HanOS

0.1.1.220618

THINK BIG, DO SMALL

Chapter 1

Todo List

File kmalloc.c

Memory allocation should be improved for better efficiency.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 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

4.2 acpi_sdt_hdr_t Struct Reference

Data Fields

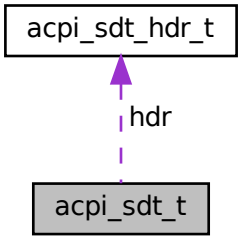
- char **sign** [4]
- uint32_t **length**
- uint8_t **rev**
- uint8_t **chksum**
- char **oem_id** [6]
- char **oem_table_id** [8]
- uint32_t **oem_rev**
- uint32_t **creator_id**
- uint32_t **creator_rev**

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

- kernel/core/acpi.h

4.3 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

4.4 addrspace_t Struct Reference

Data Fields

- uint64_t * **PML4**

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

- kernel/core/mm.h

4.5 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

4.6 command_t Struct Reference

Data Fields

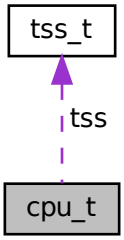
- char **command** [128]
- command_proc_t **proc**
- char **desc** [128]

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

- kernel/lib/shell.h

4.7 cpu_t Struct Reference

Collaboration diagram for cpu_t:



Data Fields

- tss_t **tss**
- uint16_t **cpu_id**
- uint16_t **lapic_id**
- bool **is_bsp**
- uint8_t **reserved** [3]

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

- kernel/core/smp.h

4.8 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

4.9 fat32_bs_info_t Struct Reference

Data Fields

- uint16_t **bytes_per_sector**
- uint8_t **sectors_per_cluster**
- uint16_t **reserved_sector_count**
- uint8_t **num_fats**
- uint32_t **sectors_per_fat**
- uint32_t **root_dir_first_cluster**
- uint32_t **fat_begin_lba**
- uint32_t **cluster_begin_lba**
- uint32_t **total_sectors**

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

- kernel/fs/fat32.h

4.10 fat32_entry_t Struct Reference

Data Fields

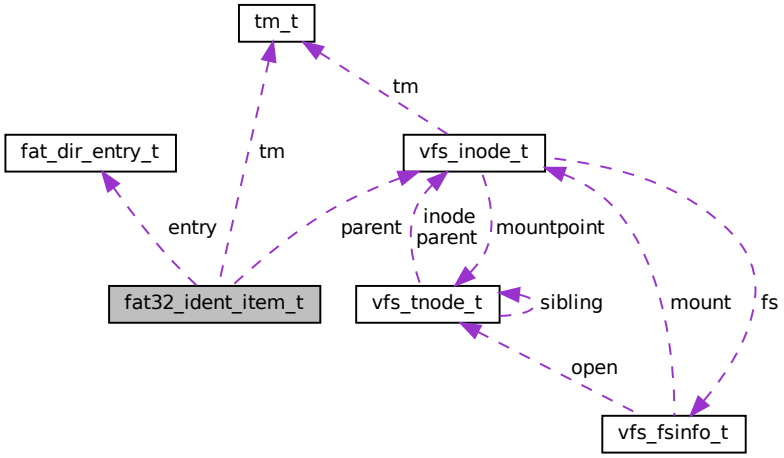
- uint8_t **name** [11]
- uint8_t **attribute**
- uint32_t **cluster_begin**
- uint32_t **file_size_bytes**
- uint32_t **dir_entry_cluster**
- size_t **dir_entry_index**

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

- kernel/fs/fat32.h

4.11 fat32_ident_item_t Struct Reference

Collaboration diagram for fat32_ident_item_t:



Data Fields

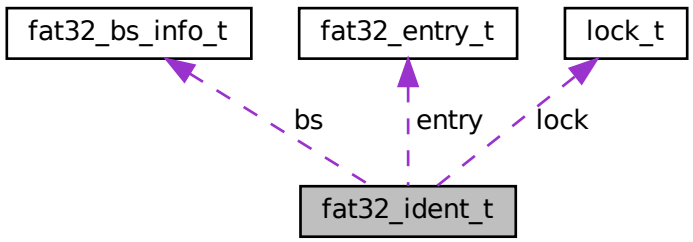
- fat_dir_entry_t **entry**
- tm_t **tm**
- char **name** [VFS_MAX_NAME_LEN]
- vfs_inode_t * **parent**

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

- kernel/fs/fat32.h

4.12 fat32_ident_t Struct Reference

Collaboration diagram for fat32_ident_t:



Data Fields

- ata_device_t * **device**
- lock_t **lock**
- fat32_bs_info_t **bs**
- fat32_entry_t **entry**
- uint32_t * **fat**
- size_t **fat_len**

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

- kernel/fs/fat32.h

4.13 fat_bs_t Struct Reference

Data Fields

- uint8_t **bootjmp** [3]
- uint8_t **oem_name** [8]
- uint16_t **bytes_per_sector**
- uint8_t **sectors_per_cluster**
- uint16_t **reserved_sector_count**
- uint8_t **table_count**
- uint16_t **root_entry_count**
- uint16_t **total_sectors_16**
- uint8_t **media_type**
- uint16_t **table_size_16**
- uint16_t **sectors_per_track**
- uint16_t **head_side_count**
- uint32_t **hidden_sector_count**
- uint32_t **total_sectors_32**
- uint8_t **extended_section** [54]

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

- kernel/fs/fat32.h

4.14 fat_dir_entry_t Struct Reference

Data Fields

- char **file_name_and_ext** [8+3]
- uint8_t **attribute**
- uint8_t **file_data_1** [2]
- uint16_t **create_time**
- uint16_t **create_date**
- uint16_t **last_visit_date**
- uint16_t **cluster_num_high**
- uint16_t **modify_time**
- uint16_t **modify_date**
- uint16_t **cluster_num_low**
- uint32_t **file_size_bytes**

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

- kernel/fs/fat32.h

4.15 fat_extbs_16_t Struct Reference

Data Fields

- uint8_t **bios_drive_num**
- uint8_t **reserved1**
- uint8_t **boot_signature**
- uint32_t **volume_id**
- uint8_t **volume_label** [11]
- uint8_t **fat_type_label** [8]

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

- kernel/fs/fat32.h

4.16 fat_extbs_32_t Struct Reference

Data Fields

- uint32_t **table_size_32**
- uint16_t **extended_flags**
- uint16_t **fat_version**
- uint32_t **root_cluster**
- uint16_t **fat_info**
- uint16_t **backup_BS_sector**
- uint8_t **reserved_0** [12]
- uint8_t **drive_number**
- uint8_t **reserved_1**
- uint8_t **boot_signature**
- uint32_t **volume_id**
- uint8_t **volume_label** [11]
- uint8_t **fat_type_label** [8]

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

- kernel/fs/fat32.h

4.17 fat_lfn_entry_t Struct Reference

Data Fields

- uint8_t **sequence_number**
- char **name1** [10]
- uint8_t **attribute**
- uint8_t **type**
- uint8_t **dos_checksum**
- char **name2** [12]
- uint16_t **first_cluster**
- char **name3** [4]

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

- kernel/fs/fat32.h

4.18 gdt_entry_t Struct Reference

Data Fields

- uint16_t **limit**
- uint16_t **base_low**
- uint8_t **base_mid**
- uint8_t **access**
- uint8_t **granularity**
- uint8_t **base_high**

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

- kernel/core/gdt.h

4.19 gdt_register_t Struct Reference

Data Fields

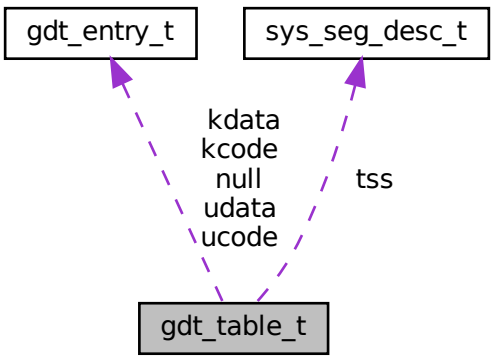
- uint16_t **size**
- uint64_t **offset**

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

- kernel/core/gdt.h

4.20 gdt_table_t Struct Reference

Collaboration diagram for gdt_table_t:



Data Fields

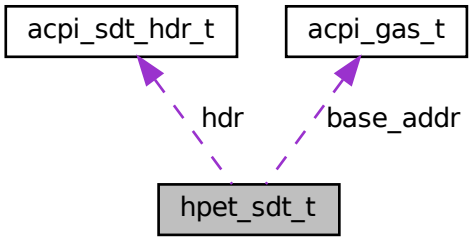
- gdt_entry_t **null**
- gdt_entry_t **kcode**
- gdt_entry_t **kdata**
- gdt_entry_t **ucode**
- gdt_entry_t **udata**
- sys_seg_desc_t **tss**

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

- kernel/core/gdt.h

4.21 hpet_sdt_t Struct Reference

Collaboration diagram for hpet_sdt_t:



Data Fields

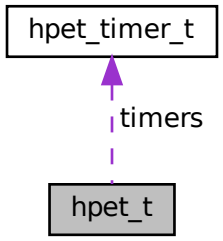
- acpi_sdt_hdr_t **hdr**
- uint8_t **hardware_rev_id**
- uint8_t **comparator_count**: 5
- uint8_t **counter_size**: 1
- uint8_t **reserved**: 1
- uint8_t **legacy_replace**: 1
- uint16_t **pci_vendor_id**
- acpi_gas_t **base_addr**
- uint8_t **hpet_number**
- uint16_t **minimum_tick**
- uint8_t **page_protection**

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

- kernel/core/hpet.h

4.22 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

4.23 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

4.24 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

4.25 idt_register_t Struct Reference

Data Fields

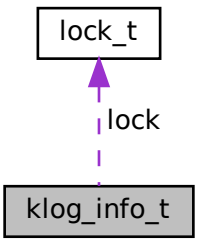
- uint16_t **size**
- uint64_t **offset**

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

- kernel/core/idt.h

4.26 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

4.27 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

4.28 madt_record_hdr_t Struct Reference

Data Fields

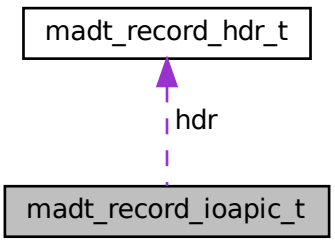
- uint8_t **type**
- uint8_t **len**

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

- kernel/core/madt.h

4.29 madt_record_ioapic_t Struct Reference

Collaboration diagram for madt_record_ioapic_t:



Data Fields

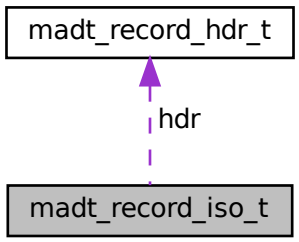
- madt_record_hdr_t **hdr**
- uint8_t **id**
- uint8_t **reserved**
- uint32_t **addr**
- uint32_t **gsi_base**

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

- kernel/core/madt.h

4.30 madt_record_iso_t Struct Reference

Collaboration diagram for madt_record_iso_t:



Data Fields

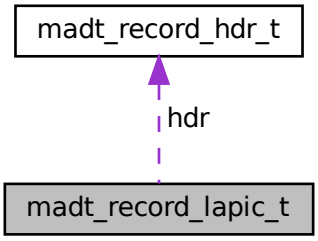
- madt_record_hdr_t **hdr**
- uint8_t **bus_src**
- uint8_t **irq_src**
- uint32_t **gsi**
- uint16_t **flags**

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

- kernel/core/madt.h

4.31 madt_record_lapic_t Struct Reference

Collaboration diagram for madt_record_lapic_t:



Data Fields

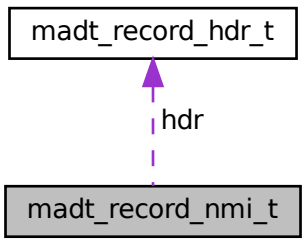
- madt_record_hdr_t **hdr**
- uint8_t **proc_id**
- uint8_t **apic_id**
- uint32_t **flags**

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

- kernel/core/madt.h

4.32 madt_record_nmi_t Struct Reference

Collaboration diagram for madt_record_nmi_t:



Data Fields

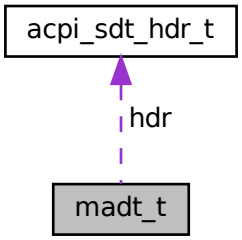
- madt_record_hdr_t **hdr**
- uint8_t **proc_id**
- uint16_t **flags**
- uint8_t **lint**

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

- kernel/core/madt.h

4.33 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

4.34 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

4.35 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

4.36 pci_device_desc_t Struct Reference

Data Fields

- uint16_t **vendor_id**
- uint16_t **device_id**
- char **desc** [256]

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

- kernel/core/pci.h

4.37 pci_device_t Struct Reference

Data Fields

- int64_t **parent**
- uint8_t **bus**
- uint8_t **func**
- uint8_t **device**
- uint16_t **device_id**
- uint16_t **vendor_id**
- uint8_t **rev_id**
- uint8_t **subclass**
- uint8_t **device_class**
- uint8_t **prog_if**
- int **multifunction**
- uint8_t **irq_pin**
- int **has_prt**
- uint32_t **gsi**
- uint16_t **gsi_flags**

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

- kernel/core/pci.h

4.38 rsdp_t Struct Reference

Data Fields

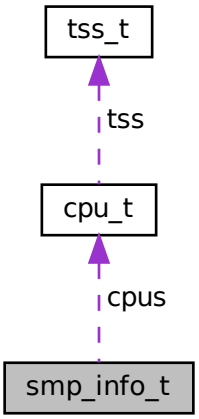
- char **sign** [8]
- uint8_t **chksum**
- char **oem_id** [6]
- uint8_t **revision**
- uint32_t **rsdt_addr**
- uint32_t **length**
- uint64_t **xsdt_addr**
- uint8_t **chksum_ext**
- uint8_t **reserved** [3]

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

- kernel/core/acpi.h

4.39 smp_info_t Struct Reference

Collaboration diagram for smp_info_t:



Data Fields

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

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

- kernel/core/smp.h

4.40 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

4.41 sys_seg_desc_t Struct Reference

Data Fields

- uint32_t **segment_limit_low**: 16
- uint32_t **segment_base_low**: 16
- uint32_t **segment_base_mid**: 8
- uint32_t **segment_type**: 4
- uint32_t **zero_1**: 1
- uint32_t **segment_dpl**: 2
- uint32_t **segment_present**: 1
- uint32_t **segment_limit_high**: 4
- uint32_t **segment_avail**: 1
- uint32_t **zero_2**: 2
- uint32_t **segment_gran**: 1
- uint32_t **segment_base_mid2**: 8
- uint32_t **segment_base_high**: 32
- uint32_t **reserved_1**: 8
- uint32_t **zero_3**: 5
- uint32_t **reserved_2**: 19

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

- kernel/core/gdt.h

4.42 task_regs_t Struct Reference

Data Fields

- uint64_t **r15**
- uint64_t **r14**
- uint64_t **r13**
- uint64_t **r12**
- uint64_t **r11**
- uint64_t **r10**
- uint64_t **r9**
- uint64_t **r8**

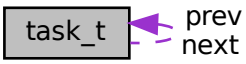
- uint64_t **rbp**
- uint64_t **rdi**
- uint64_t **rsi**
- uint64_t **rdx**
- uint64_t **rcx**
- uint64_t **rbx**
- uint64_t **rax**
- uint64_t **rip**
- uint64_t **cs**
- uint64_t **rflags**
- uint64_t **rsp**
- uint64_t **ss**

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

- kernel/proc/task.h

4.43 task_t Struct Reference

Collaboration diagram for task_t:



Data Fields

- void * **kstack_top**
- void * **kstack_limit**
- void * **ustack_top**
- void * **ustack_limit**
- task_id_t **tid**
- task_priority_t **priority**
- uint64_t **last_tick**
- uint64_t **wakeup_time**
- task_status_t **status**
- task_mode_t **mode**
- struct task_t * **next**
- struct task_t * **prev**

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

- kernel/proc/task.h

4.44 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

4.45 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

4.46 tss_t Struct Reference

Data Fields

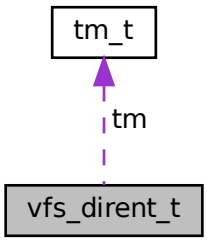
- uint32_t **reserved**
- uint64_t **rsp0**
- uint64_t **rsp1**
- uint64_t **rsp2**
- uint32_t **reserved_1**
- uint32_t **reserved_2**
- uint64_t **ist1**
- uint64_t **ist2**
- uint64_t **ist3**
- uint64_t **ist4**
- uint64_t **ist5**
- uint64_t **ist6**
- uint64_t **ist7**
- uint64_t **reserved_3**
- uint16_t **reserved_4**
- uint16_t **io_bitmap_offset**

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

- kernel/core/smp.h

4.47 vfs_dirent_t Struct Reference

Collaboration diagram for vfs_dirent_t:



Data Fields

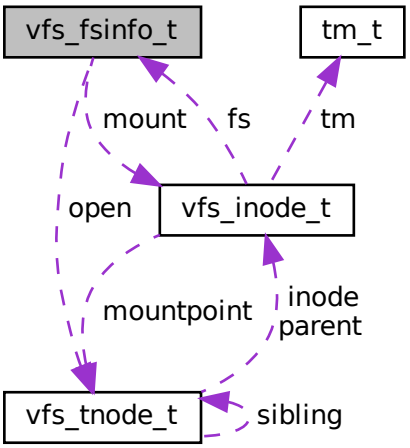
- vfs_node_type_t **type**
- tm_t **tm**
- char **name** [VFS_MAX_NAME_LEN]

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

- kernel/fs/vfs.h

4.48 vfs_fsinfo_t Struct Reference

Collaboration diagram for vfs_fsinfo_t:



Public Member Functions

- **vec_struct** (void ∗) filelist

Data Fields

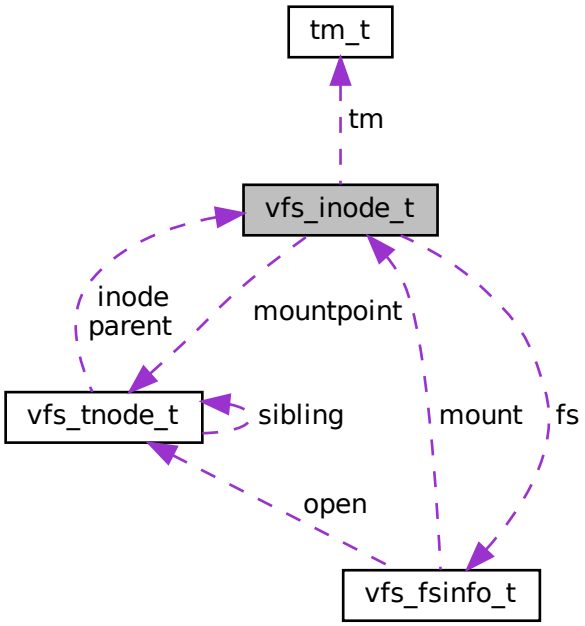
- char **name** [16]
- bool **istemp**
- vfs_inode_t ∗(∗ **mount**)(vfs_inode_t ∗device)
- vfs_tnode_t ∗(∗ **open**)(vfs_inode_t ∗this, const char ∗path)
- int64_t(∗ **mknode**)(vfs_tnode_t ∗this)
- int64_t(∗ **read**)(vfs_inode_t ∗this, size_t offset, size_t len, void ∗buff)
- int64_t(∗ **write**)(vfs_inode_t ∗this, size_t offset, size_t len, const void ∗buff)
- int64_t(∗ **sync**)(vfs_inode_t ∗this)
- int64_t(∗ **refresh**)(vfs_inode_t ∗this)
- int64_t(∗ **getdent**)(vfs_inode_t ∗this, size_t pos, vfs_dirent_t ∗dirent)
- int64_t(∗ **ioctl**)(vfs_inode_t ∗this, int64_t req_param, void ∗req_data)

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

- kernel/fs/vfs.h

4.49 vfs_inode_t Struct Reference

Collaboration diagram for vfs_inode_t:



Public Member Functions

- **vec_struct** (vfs_tnode_t ∗) child

Data Fields

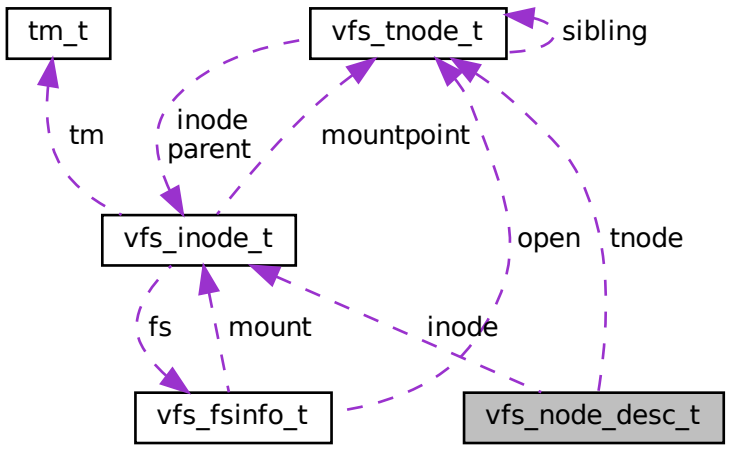
- vfs_node_type_t **type**
- size_t **size**
- uint32_t **perms**
- uint32_t **uid**
- uint32_t **refcount**
- tm_t **tm**
- vfs_fsinfo_t ∗ **fs**
- void ∗ **ident**
- vfs_tnode_t ∗ **mountpoint**

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

- kernel/fs/vfs.h

4.50 vfs_node_desc_t Struct Reference

Collaboration diagram for vfs_node_desc_t:



Data Fields

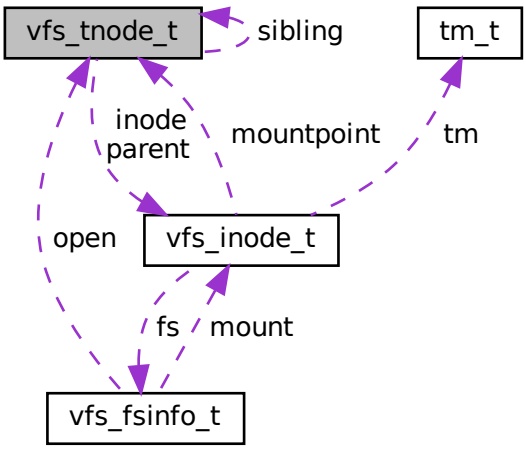
- char **path** [VFS_MAX_PATH_LEN]
- vfs_tnode_t * **tnode**
- vfs_inode_t * **inode**
- vfs_openmode_t **mode**
- size_t **seek_pos**

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

- kernel/fs/vfs.h

4.51 vfs_tnode_t Struct Reference

Collaboration diagram for vfs_tnode_t:



Data Fields

- char **name** [VFS_MAX_NAME_LEN]
- vfs_inode_t * **inode**
- vfs_inode_t * **parent**
- vfs_tnode_t * **sibling**

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

- kernel/fs/vfs.h

Chapter 5

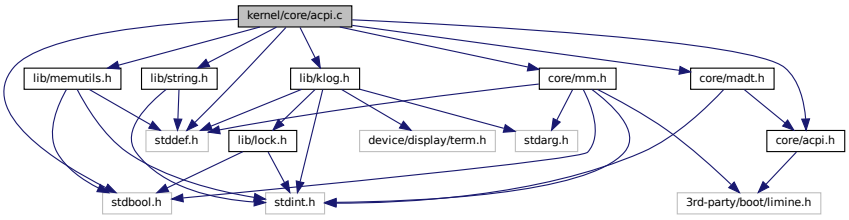
File Documentation

5.1 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 <lib/string.h>
```

Include dependency graph for acpi.c:



Functions

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

5.1.1 Detailed Description

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

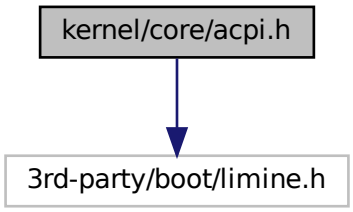
```
This module includes implementation of RSDT/XSDT initialization and
"MADT/HPET" parsing.
```

5.2 kernel/core/acpi.h File Reference

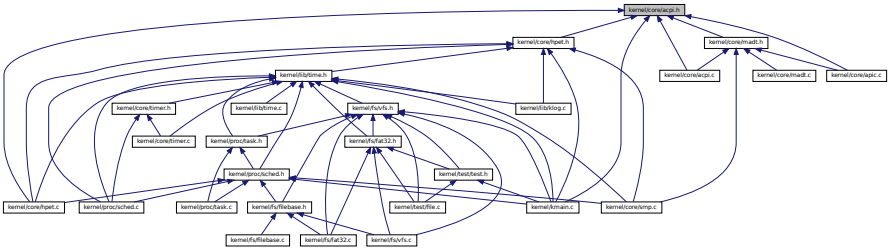
Definition of ACPI (Advanced Configuration and Power Management Interface) related data structures.

`#include <3rd-party/boot/limine.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 limine_rsdp_response *)
- acpi_sdt_t * **acpi_get_sdt** (const char *sign)

5.2.1 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
```

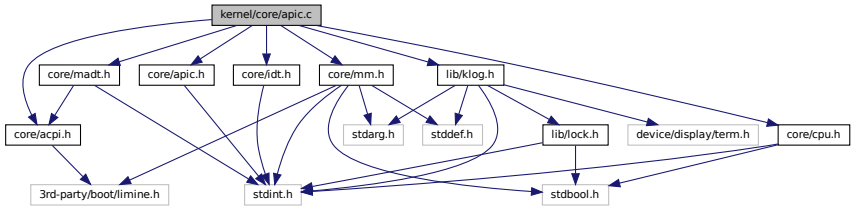
```
time ACPI environment, which consists of AML code (a platform independent
OOP language that comes from the BIOS and devices) and the ACPI SMM
(System Management Mode) code.
```

5.3 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** ()

5.3.1 Detailed Description

Implementation of APIC (Advanced Programmable Interrupt Controller) functions.

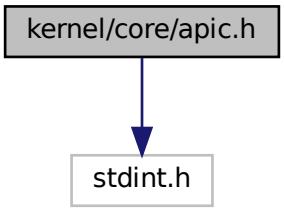
```
APIC ("Advanced Programmable Interrupt Controller") is the updated Intel
standard for the older PIC. It is used in multiprocessor systems and is
an integral part of all recent Intel (and compatible) processors. The
APIC is used for sophisticated interrupt redirection, and for sending
interrupts between processors.

Ref: https://wiki.osdev.org/APIC
```

5.4 kernel/core/apic.h File Reference

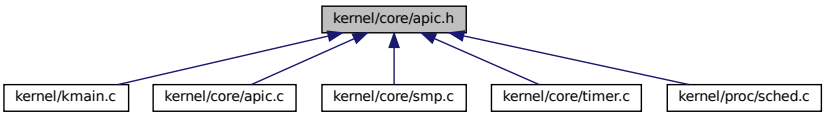
Definition of APIC (Advanced Programmable Interrupt Controller) related data structures.

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

5.4.1 Detailed Description

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

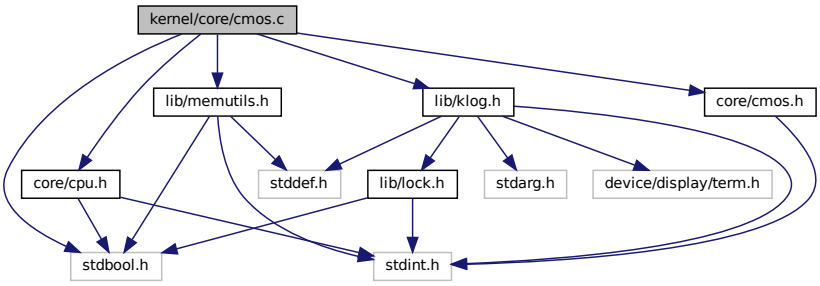
```
APIC is used in multiprocessor systems and is an integral part of all
recent Intel (and compatible) processors. The APIC is used for
sophisticated interrupt redirection, and for sending interrupts between
processors.
```

5.5 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:



Macros

- #define **CURRENT_YEAR** 2022 /∗ Change this each year! ∗/

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)
- void **cmos_init** ()
- uint64_t **cmos_boot_time** ()
- uint64_t **cmos_current_time** ()
- cmos_rtc_t **cmos_read_rtc** ()
- uint64_t **secs_of_years** (uint64_t years)
- uint64_t **secs_of_month** (uint64_t months, uint64_t year)

Variables

- int **century_register** = 0x00

5.5.1 Detailed Description

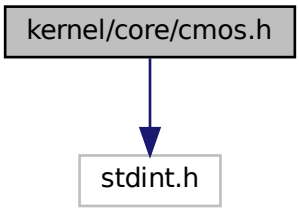
Implementation of CMOS related functions.

```
"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.
```

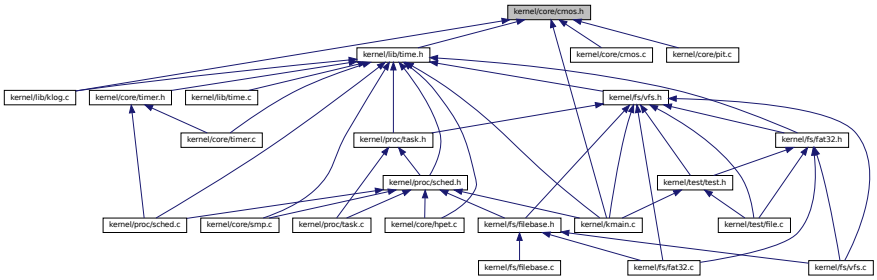
Ref: https://wiki.osdev.org/CMOS

5.6 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

- uint64_t **secs_of_month** (uint64_t months, uint64_t year)
- uint64_t **secs_of_years** (uint64_t years)
- void **cmos_init** ()
- cmos_rtc_t **cmos_read_rtc** ()
- uint64_t **cmos_boot_time** ()
- uint64_t **cmos_current_time** ()

5.6.1 Detailed Description

Definition of CMOS related data structures.

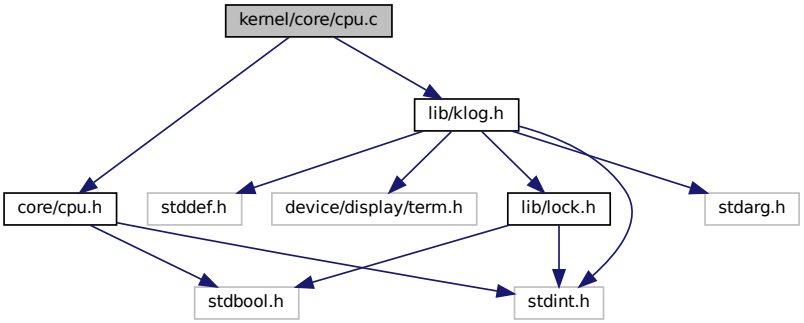
```
"CMOS" is a tiny bit of very low power static memory that lives on the
same chip as the Real-Time Clock (RTC). It was introduced to IBM PC AT in
1984 which used Motorola MC146818A RTC.
```

5.7 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** ()
- bool **cpu_ok** ()

5.7.1 Detailed Description

Implementation of CPU related functions.

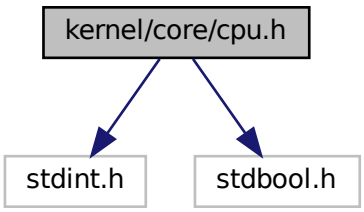
```
e.g., CPU initialization...
```

5.8 kernel/core/cpu.h File Reference

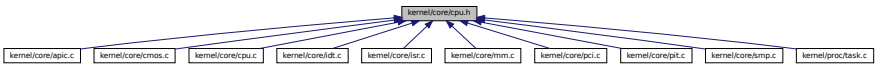
Definition of CPU related data structures and macros.

```
#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 **cpu_ok** ()
- bool **cpuid_check_feature** (cpuid_feature_t feature)

5.8.1 Detailed Description

Definition of CPU related data structures and macros.

```
e.g., Read & write control registers, model specific registers and port
input & output.
```

5.8.2 Macro Definition Documentation

5.8.2.1 read_cr

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

Value:

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

5.8.2.2 write_cr

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

Value:

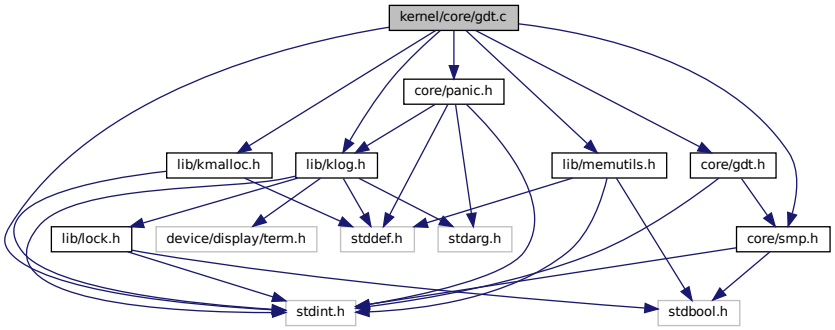
```
asm volatile("mov %0, %%rax;"       \
"mov %%rax, %%" cr       \
:                        \
: "g"(n)                 \
: "rax");
```

5.9 kernel/core/gdt.c File Reference

Implementation of GDT related functions.

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

Include dependency graph for gdt.c:



Functions

- void **gdt_init** (cpu_t *cpuinfo)
- void **gdt_install_tss** (cpu_t *cpuinfo)

5.9.1 Detailed Description

Implementation of GDT related functions.

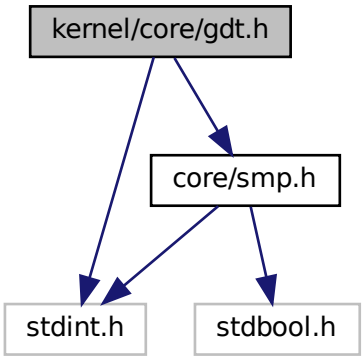
```
The Global Descriptor Table (GDT) contains entries telling the CPU about
memory segments.

In HanOS, GDT initialization is very simple. Only memory protection is
used. Two ring-0 and two ring-3 segment descriptor are defined. The
memory regions are from 0 to 4GB.
```

5.10 kernel/core/gdt.h File Reference

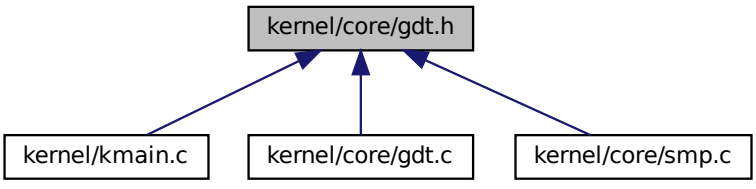
Definition of GDT related data structures.

```
#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 gdt_entry_t
- struct sys_seg_desc_t
- struct gdt_table_t
- struct gdt_register_t

Functions

- void **gdt_init** (cpu_t *cpuinfo)
- void **gdt_install_tss** (cpu_t *cpuinfo)

5.10.1 Detailed Description

Definition of GDT related data structures.

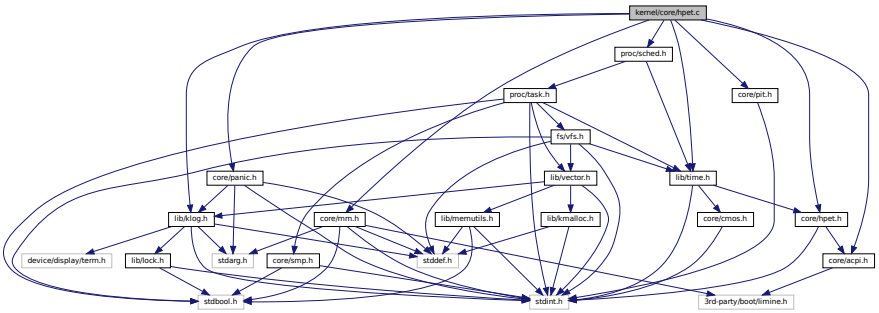
```
The Global Descriptor Table (GDT) contains entries telling the CPU about
memory segments.
```

5.11 kernel/core/hpet.c File Reference

Implementation of HPET (High Precision Event Timer) functions.

```
#include <lib/klog.h>
#include <lib/time.h>
#include <core/hpet.h>
#include <core/acpi.h>
#include <core/mm.h>
#include <core/panic.h>
#include <core/pit.h>
#include <proc/sched.h>
```

Include dependency graph for hpet.c:



Functions

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

5.11.1 Detailed Description

Implementation of HPET (High Precision Event Timer) functions.

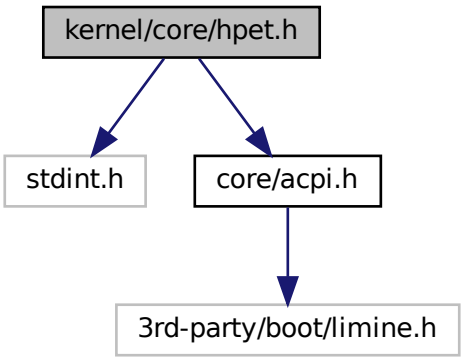
```
This module includes implementation of HPET initialization, getting nanos
and nanos sleep.

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.
```

5.12 kernel/core/hpet.h File Reference

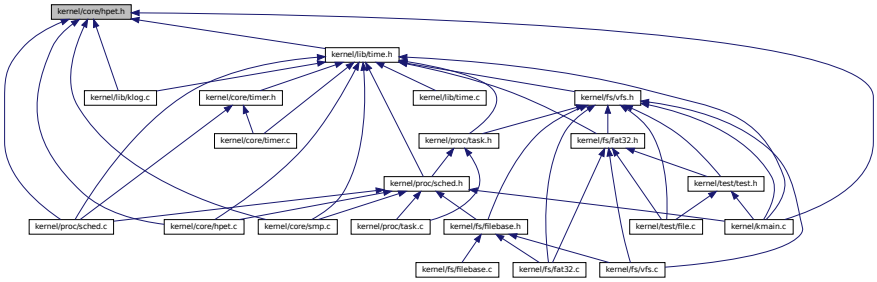
Definition of HPET (High Precision Event Timer) related data structures.

```
#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)

5.12.1 Detailed Description

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

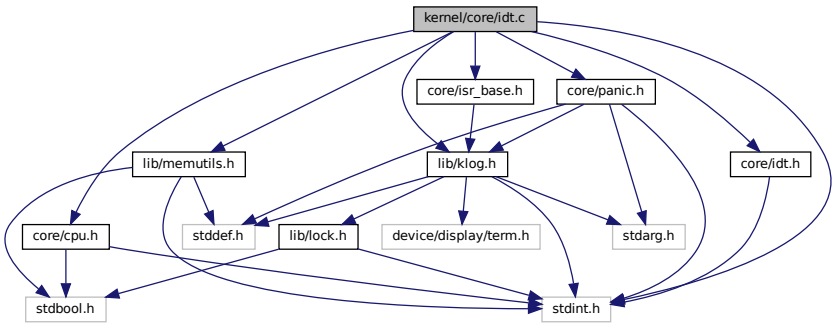
```
HPET, or High Precision Event Timer, is a piece of hardware designed by
Intel and Microsoft to replace older PIT and RTC. It consists of (usually
64-bit) main counter (which counts up), as well as from 3 to 32 32-bit or
64-bit wide comparators. HPET is programmed using memory mapped IO, and
the base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

5.13 kernel/core/idt.c File Reference

Implementation of idt related functions.

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

Include dependency graph for idt.c:



Functions

- void **idt_set_handler** (uint8_t vector, void ∗handler)
- uint8_t **idt_get_available_vector** (void)
- void **irq_set_mask** (uint8_t line)
- void **irq_clear_mask** (uint8_t line)
- void **idt_init** ()

5.13.1 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.
```

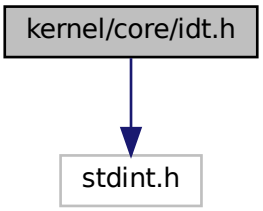
Ref: https://wiki.osdev.org/Interrupt_Descriptor_Table

5.14 kernel/core/idt.h File Reference

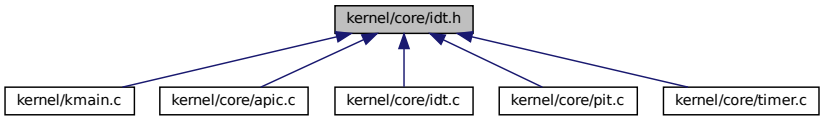
Definition of IDT related data structures.

```
#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** ()
- void **idt_set_handler** (uint8_t vector, void ∗handler)
- uint8_t **idt_get_available_vector** (void)
- void **irq_set_mask** (uint8_t line)
- void **irq_clear_mask** (uint8_t line)

5.14.1 Detailed Description

Definition of IDT related data structures.

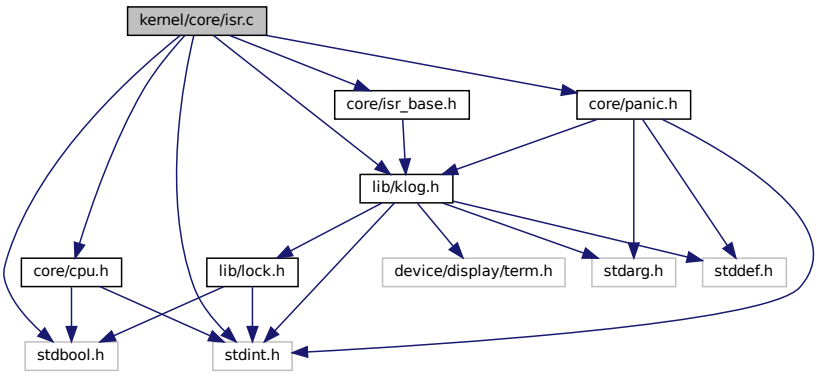
```
The Interrupt Descriptor Table (IDT) telling the CPU where the Interrupt
Service Routines (ISR) are located (one per interrupt vector).
```

5.15 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 <core/cpu.h>
```

Include dependency graph for isr.c:



Functions

- void **exc_register_handler** (uint64_t id, exc_handler_t handler)
- void **exc_handler_proc** (uint64_t errcode, uint64_t excno)

5.15.1 Detailed Description

Implementation of ISR related functions.

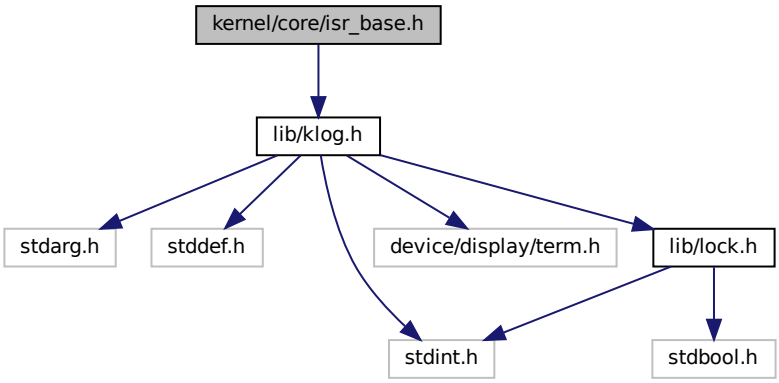
```
The x86 architecture is an interrupt driven system. Only a common interrupt
handling function is implemented.
```

5.16 kernel/core/isr_base.h File Reference

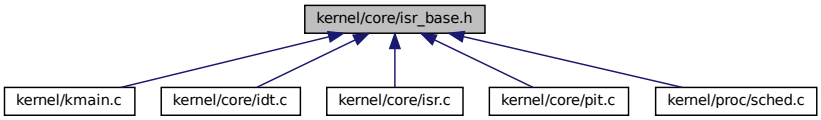
Definition of ISR related data structures.

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

Include dependency graph for isr_base.h:



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



Macros

- #define **PIC1** 0x20 /∗ Master PIC ∗/
- #define **PIC2** 0xA0 /∗ Slave PIC ∗/
- #define **PIC1_DATA** (PIC1 + 1)
- #define **PIC2_DATA** (PIC2 + 1)
- #define **PIC_EOI** 0x20 /∗ end of interrupt ∗/
- #define **IRQ_BASE** 0x20
- #define **IRQ0** 32
- #define **IRQ1** 33
- #define **IRQ2** 34
- #define **IRQ3** 35
- #define **IRQ4** 36
- #define **IRQ5** 37
- #define **IRQ6** 38
- #define **IRQ7** 39
- #define **IRQ8** 40
- #define **IRQ9** 41
- #define **IRQ10** 42
- #define **IRQ11** 43
- #define **IRQ12** 44
- #define **isr_enable_interrupts**()
- #define **isr_disable_interrupts**()

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)
- void **irq0** (void ∗p)
- void **irq1** (void ∗p)
- void **irq2** (void ∗p)
- void **irq3** (void ∗p)
- void **irq4** (void ∗p)
- void **irq5** (void ∗p)
- void **irq6** (void ∗p)
- void **irq7** (void ∗p)
- void **irq8** (void ∗p)
- void **irq9** (void ∗p)
- void **irq10** (void ∗p)
- void **irq11** (void ∗p)
- void **irq12** (void ∗p)

5.16.1 Detailed Description

Definition of ISR related data structures.

```
The x86 architecture is an interrupt driven system.
```

5.16.2 Macro Definition Documentation

5.16.2.1 isr_disable_interrupts

```
#define isr_disable_interrupts( )
```

Value:

```
{                                       \
    asm volatile("cli");                \
}
```

5.16.2.2 isr_enable_interrupts

```
#define isr_enable_interrupts( )
```

Value:

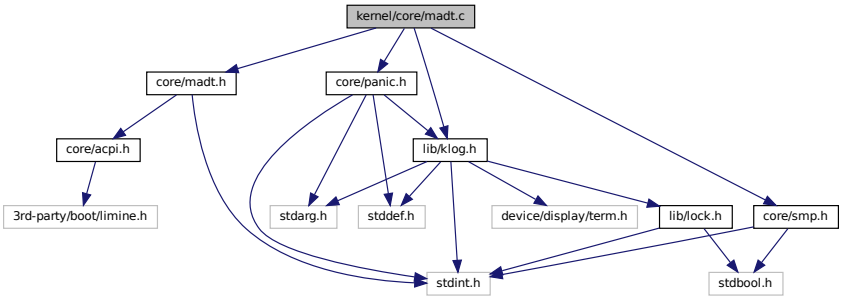
```
    {                                           \
        asm volatile("sti");                    \
    }
```

5.17 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** ()

5.17.1 Detailed Description

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

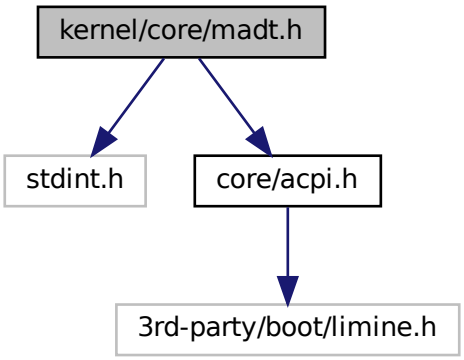
```
The MADT describes all of the interrupt controllers in the system. It
can be used to enumerate the processors currently available.
```

5.18 kernel/core/madt.h File Reference

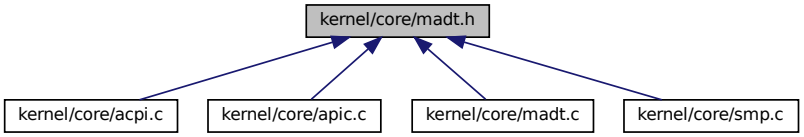
Definition of ACPI MADT (Multiple APIC Description Table) related data structures.

```
#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** ()

5.18.1 Detailed Description

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

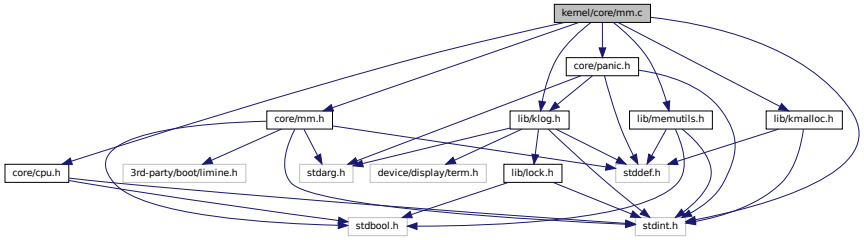
```
The MADT describes all of the interrupt controllers in the system. It
can be used to enumerate the processors currently available.
```

5.19 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, uint64_t baseaddr)
- void **pmm_init** (struct limine_memmap_response ∗map)
- void **pmm_dump_usage** (void)
- 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** (struct limine_memmap_response ∗map, struct limine_kernel_address_response ∗kernel)

5.19.1 Detailed Description

Implementation of memory management functions.

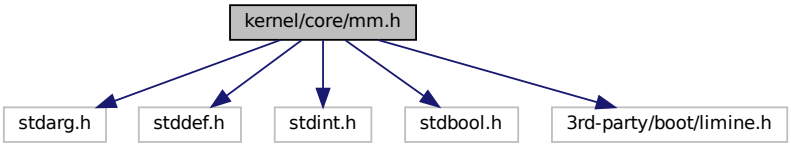
```
Memory management is a critical part of any operating system kernel.
Providing a quick way for programs to allocate and free memory on a
regular basis is a major responsibility of the kernel.

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.
```

5.20 kernel/core/mm.h File Reference

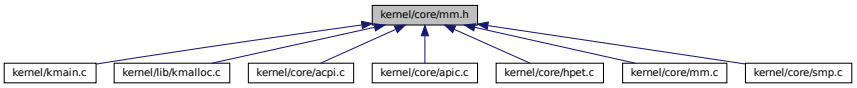
Definition of memory management related data structures.

```
#include <stdarg.h>
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <3rd-party/boot/limine.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 **USERSPACE_OFFSET** 0x1000
- #define **HIGHERHALF_OFFSET** 0xffffffff80000000
- #define **KERNEL_OFFSET** 0xffffffff80200000

- #define **NUM_PAGES**(num) (((num) + PAGE_SIZE - 1) / PAGE_SIZE)
- #define **PAGE_ALIGN_UP**(num) (NUM_PAGES(num) ∗ PAGE_SIZE)
- #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 **MEM_VIRT_OFFSET** 0xffff800000000000
- #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 limine_memmap_response ∗map)
- uint64_t **pmm_get** (uint64_t numpages, uint64_t baseaddr)
- bool **pmm_alloc** (uint64_t addr, uint64_t numpages)
- void **pmm_free** (uint64_t addr, uint64_t numpages)
- void **pmm_dump_usage** (void)
- void **vmm_init** (struct limine_memmap_response ∗map, struct limine_kernel_address_response ∗kernel)
- void **vmm_map** (uint64_t vaddr, uint64_t paddr, uint64_t np, uint64_t flags)
- void **vmm_unmap** (uint64_t vaddr, uint64_t np)

5.20.1 Detailed Description

Definition of memory management related data structures.

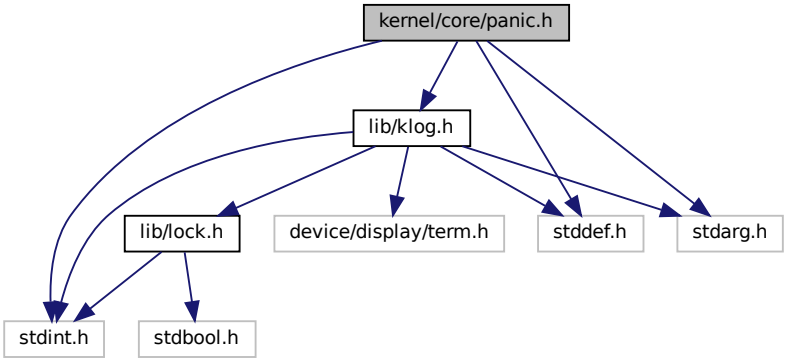
```
Memory management is a critical part of any operating system kernel.
Providing a quick way for programs to allocate and free memory on a
regular basis is a major responsibility of the kernel.
```

5.21 kernel/core/panic.h File Reference

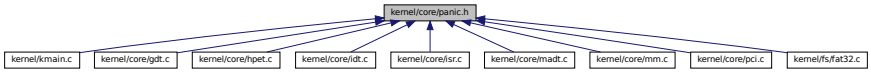
Implementation of panic related functions.

```
#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, ...)
- #define **panic_unless**(c)
- #define **panic_if**(c)

Functions

- void **dump_backtrace** ()

5.21.1 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.
```

5.21.2 Macro Definition Documentation

5.21.2.1 kpanic

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

Value:

```
    { \
    asm volatile("cli"); \
    klog_vprintf(KLOG_LEVEL_ERROR, s, ##__VA_ARGS__); \
```

```
    dump_backtrace(); \
    while (true) \
        asm volatile("hlt"); \
}
```

5.21.2.2 panic_if

```
#define panic_if(
                c )
```

Value:

```
    ({ \
    if((c)) \
        kpanic("panic_if(" #c ") triggered in " \
              "__FILE__:%d", __LINE__); \
})
```

5.21.2.3 panic_unless

```
#define panic_unless(
                c )
```

Value:

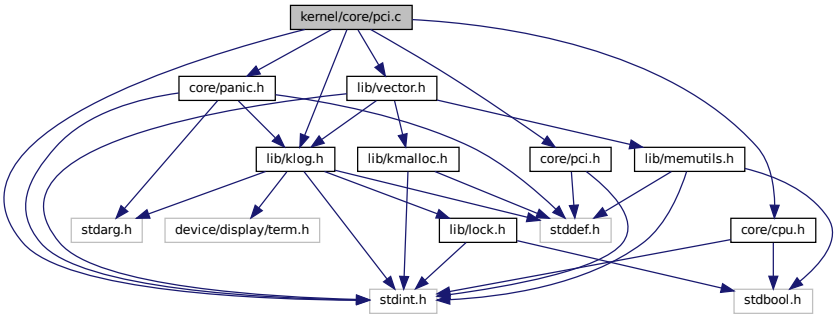
```
    ({ \
    if(!(c)) \
        kpanic("panic_unless(" #c ") triggered in " \
              "__FILE__:%d", __LINE__);     \
})
```

5.22 kernel/core/pci.c File Reference

Implementation of PCI related functions.

```
#include <stdint.h>
#include <core/pci.h>
#include <core/cpu.h>
#include <core/panic.h>
#include <lib/klog.h>
#include <lib/vector.h>
```

Include dependency graph for pci.c:



Macros

- #define **MAX_FUNCTION** 8
- #define **MAX_DEVICE** 16
- #define **pci_func_exist**(dev) ((uint16_t)(pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0) & 0x↩ FFFF) != 0xFFFF)

- #define **pci_read_vendor_id**(dev) (pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x00) & 0x↩ FFFF)
- #define **pci_read_device_id**(dev) (pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x00) >> 16)
- #define **pci_read_class**(dev) (pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x08) >> 24)
- #define **pci_read_subclass**(dev) ((pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x08) >> 16) & 0xFF)
- #define **pci_read_prog_if**(dev) ((pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x08) >> 8) & 0xFF)
- #define **pci_read_header**(dev) (((pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x08) >> 16) & ~(1 << 7)) & 0xFF)
- #define **pci_read_sub_bus**(dev) ((pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x18) >> 8) & 0xFF)
- #define **pci_is_bridge**(dev) (pci_read_header(dev) == 0x1 && pci_read_class(dev) == 0x6)
- #define **pci_has_multi_func**(dev) (((pci_read_dword((dev)->bus, (dev)->device, (dev)->func, 0x0C) >> 16) & (1 << 7)) & 0xFF)

Functions

- **vec_new_static** (pci_device_t, pci_devices)
- void **pci_init** (void)
- void **pci_debug** (void)

5.22.1 Detailed Description

Implementation of PCI related functions.

```
There are two functions in this file:
1. pci_init() which can scan all PCI devices and store information into an
   array.
2. pci_debug() which can be called by command line and display PCI device
   list.
```

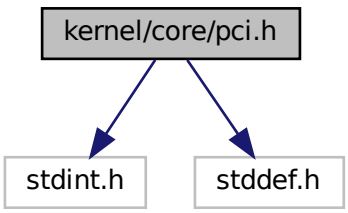
Ref: https://wiki.osdev.org/PCI

5.23 kernel/core/pci.h File Reference

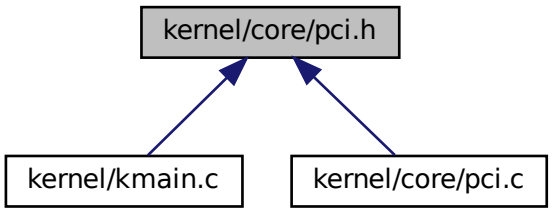
Definition of PCI related data structures and functions.

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

Include dependency graph for pci.h:



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



Data Structures

- struct pci_device_t
- struct pci_device_desc_t

Functions

- void **pci_init** (void)
- void **pci_debug** (void)

5.23.1 Detailed Description

Definition of PCI related data structures and functions.

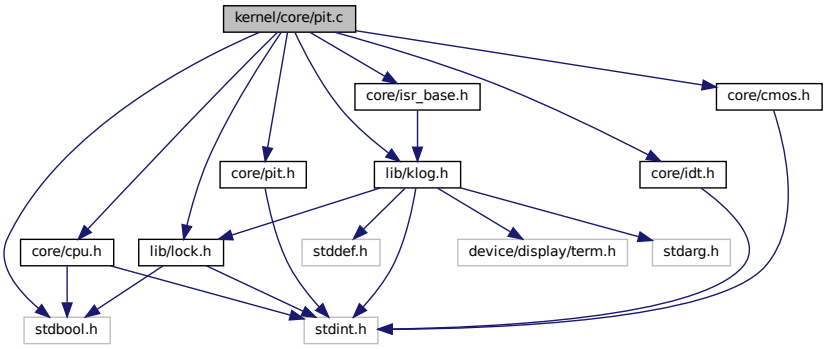
```
The PCI (Peripheral Component Interconnect) bus was defined to establish a
high performance and low cost local bus that would remain through several
generations of products.
```

5.24 kernel/core/pit.c File Reference

Implementation of PIT related functions.

```
#include <stdbool.h>
#include <core/pit.h>
#include <core/cpu.h>
#include <core/idt.h>
#include <core/cmos.h>
#include <core/isr_base.h>
#include <lib/lock.h>
#include <lib/klog.h>
```

Include dependency graph for pit.c:



Macros

- #define **PIT_FREQ_HZ** 1000

Functions

- uint64_t **pit_get_ticks** (void)
- void **pit_init** (void)
- void **pit_wait** (uint64_t ms)

5.24.1 Detailed Description

Implementation of PIT related functions.

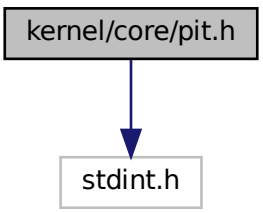
```
This PIT wait function should be rewritten based on IRQ mechanism, then it
can be used in different processes at the same time.

The PIT wait should only be used in system initialization stage. It will not
be re-written after re-considering it's usage.
```

5.25 kernel/core/pit.h File Reference

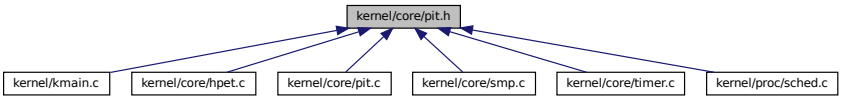
Definition of PIT related data structures.

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

Include dependency graph for pit.h:



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



Functions

- void **pit_init** (void)
- void **pit_wait** (uint64_t ms)
- uint64_t **pit_get_ticks** (void)

5.25.1 Detailed Description

Definition of PIT related data structures.

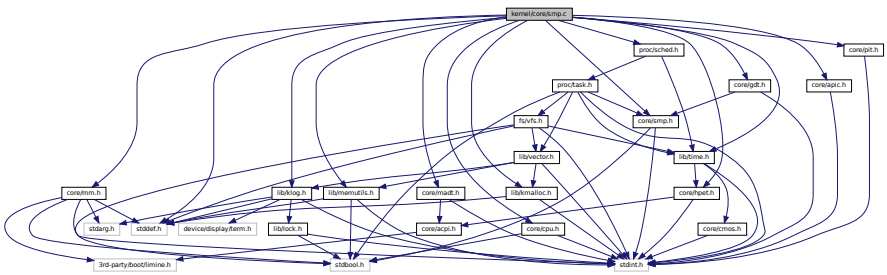
```
The Programmable Interval Timer (PIT) chip (Intel 8253/8254) basically
consists of an oscillator, a prescaler and 3 independent frequency dividers.
Each frequency divider has an output, which is used to allow the timer to
control external circuitry (for example, IRQ 0).
```

5.26 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 <core/pit.h>
#include <proc/sched.h>
```

Include dependency graph for smp.c:



Functions

- const smp_info_t * **smp_get_info** ()

- cpu_t * **smp_get_current_cpu** (bool force_read)
- void **init_tss** (cpu_t *cpuinfo)
- _Noreturn void **smp_ap_entrypoint** (cpu_t *cpuinfo)
- void **smp_init** ()

Variables

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

5.26.1 Detailed Description

Implementation of SMP related functions.

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

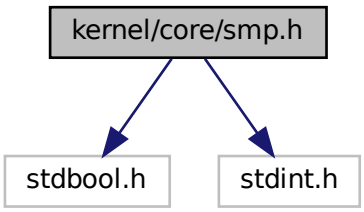
Ref: https://wiki.osdev.org/SMP

5.27 kernel/core/smp.h File Reference

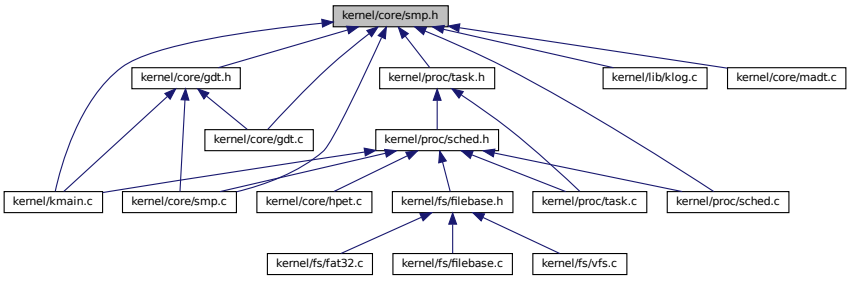
Definition of SMP related data structures.

```
#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** (void)
- const smp_info_t * **smp_get_info** (void)
- cpu_t * **smp_get_current_cpu** (bool force_read)

5.27.1 Detailed Description

Definition of SMP related data structures.

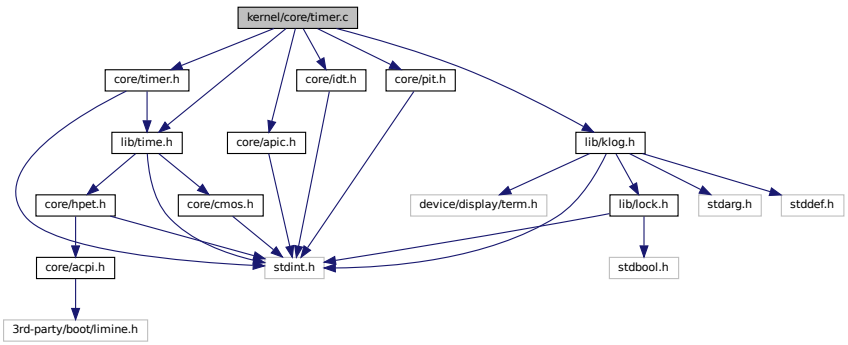
```
Symmetric Multiprocessing (or SMP) is one method of having multiple
processors in one computer system.
```

5.28 kernel/core/timer.c File Reference

Implementation of APIC timer related functions.

```
#include <core/timer.h>
#include <core/apic.h>
#include <core/idt.h>
#include <core/pit.h>
#include <lib/klog.h>
#include <lib/time.h>
```

Include dependency graph for timer.c:



Functions

- void **apic_timer_stop** (void)
- void **apic_timer_start** (void)
- void **apic_timer_set_handler** (void(∗h)(void ∗))
- void **apic_timer_set_frequency** (uint64_t freq)
- void **apic_timer_set_period** (time_t tv)
- uint8_t **apic_timer_get_vector** (void)
- void **apic_timer_set_mode** (apic_timer_mode_t mode)
- void **apic_timer_enable** (void)
- void **apic_timer_init** (void)

5.28.1 Detailed Description

Implementation of APIC timer related functions.

```
The timer has 2 or 3 modes. The first 2 modes (periodic and one-shot)
are supported by all local APICs. The third mode (TSC-Deadline mode) is
an extension that is only supported on recent CPUs.

Periodic Mode:
- Generate IRQs at a fixed rate depending on the initial count.

One-Shot Mode:
- Decrement the current count (and generates a timer IRQ when the count
  reaches zero) in the same way as in periodic mode. However it doesn't
  reset the current count to the initial count when the current count
  reaches zero.

TSC-Deadline mode:
- Similar with one-shot mode but using CPU's time stamp counter instead
  to get higher precision.
```

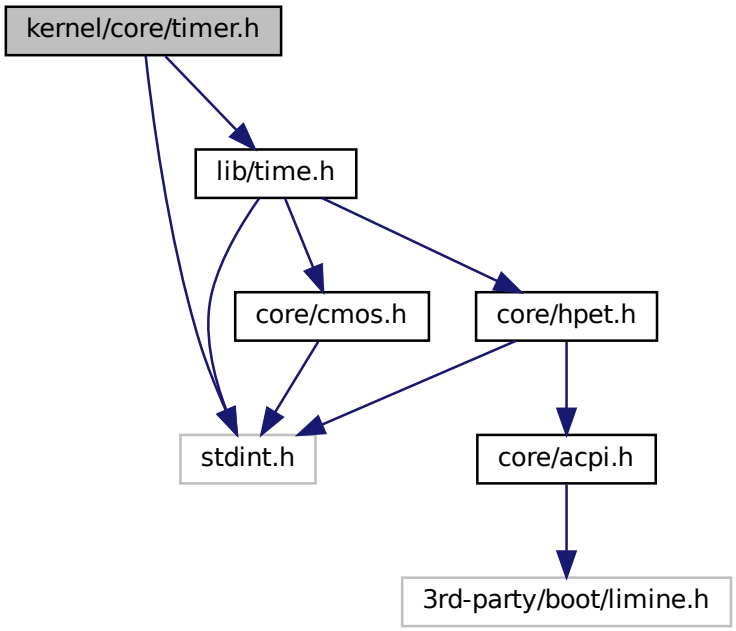
Ref: https://wiki.osdev.org/APIC_timer

5.29 kernel/core/timer.h File Reference

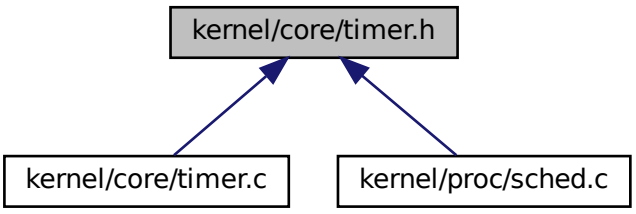
Definition of APIC timer related macros, data structures, functions.

```
#include <stdint.h>
#include <lib/time.h>
```

Include dependency graph for timer.h:



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



Macros

- #define **APIC_REG_TIMER_LVT** 0x320
- #define **APIC_REG_TIMER_ICR** 0x380
- #define **APIC_REG_TIMER_CCR** 0x390
- #define **APIC_REG_TIMER_DCR** 0x3e0
- #define **APIC_TIMER_FLAG_PERIODIC** (1 << 17)
- #define **APIC_TIMER_FLAG_MASKED** (1 << 16)

Enumerations

- enum **apic_timer_mode_t** { **APIC_TIMER_MODE_PERIODIC**, **APIC_TIMER_MODE_ONESHOT** }

Functions

- void **apic_timer_init** (void)
- void **apic_timer_enable** (void)
- void **apic_timer_stop** (void)
- void **apic_timer_start** (void)
- void **apic_timer_set_handler** (void(∗h)(void ∗))
- void **apic_timer_set_frequency** (uint64_t freq)
- void **apic_timer_set_period** (time_t tv)
- void **apic_timer_set_mode** (apic_timer_mode_t mode)
- uint8_t **apic_timer_get_vector** (void)

5.29.1 Detailed Description

Definition of APIC timer related macros, data structures, functions.

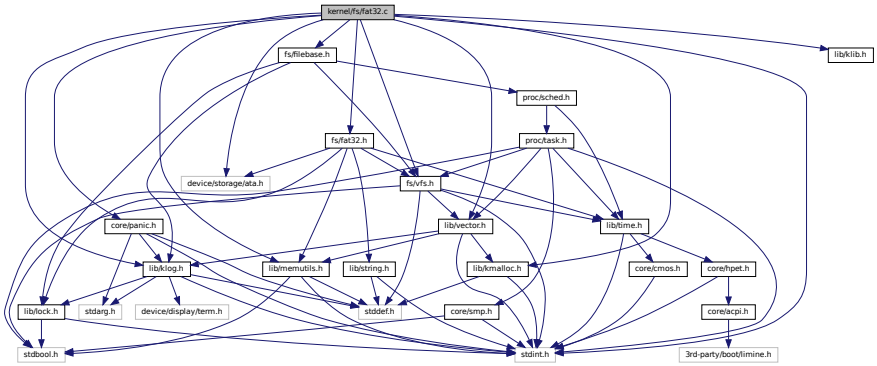
```
The great benefit of the Local APIC timer is that it is hardwired to each
CPU core, unlike the Programmable Interval Timer which is a separate
circuit. Because of this, there is no need for any resource management,
which makes things easier. The downside is that it's oscillating at (one
of) the CPU's frequencies, which varies from machine to machine, while
the PIT uses a standard frequency (1,193,182 Hz). To make use of it, you
have to know how many interrupts/sec it's capable of.
```

5.30 kernel/fs/fat32.c File Reference

Implementation of FAT32 related functions.

```
#include <stdint.h>
#include <device/storage/ata.h>
#include <core/panic.h>
#include <fs/vfs.h>
#include <fs/fat32.h>
#include <fs/filebase.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <lib/klog.h>
#include <lib/klib.h>
#include <lib/vector.h>
```

Include dependency graph for fat32.c:



Macros

- #define **SECTORSHIFT** 9 /∗ 512 bytes ∗/
- #define **SECTOR_TO_OFFSET**(x) ((x) << SECTORSHIFT)
- #define **CLUSTER_TO_OFFSET**(x, c, s) (SECTOR_TO_OFFSET((c) + ((x) - 2) ∗ (s)))

Functions

- int **fat32_read_entry** (vfs_inode_t *this, uint32_t cluster, size_t index, fat32_entry_t *dest)
- int **fat32_write_entry** (vfs_inode_t *this, fat32_entry_t *src)
- int64_t **fat32_read** (vfs_inode_t *this, size_t offset, size_t len, void *buff)
- int64_t **fat32_write** (vfs_inode_t *this, size_t offset, size_t len, const void *buff)
- int64_t **fat32_sync** (vfs_inode_t *this)
- int64_t **fat32_getdent** (vfs_inode_t *this, size_t pos, vfs_dirent_t *dirent)
- int64_t **fat32_refresh** (vfs_inode_t *this)
- int64_t **fat32_mknode** (vfs_tnode_t *this)
- int **fat32_compare_entry_and_path** (fat32_entry_t *ent, const char *path)
- void **fat32_dump_entry** (fat32_entry_t fe)
- fat32_entry_t **fat32_parse_path** (vfs_inode_t *this, const char *path)
- vfs_tnode_t * **fat32_open** (vfs_inode_t *this, const char *path)
- vfs_inode_t * **fat32_mount** (vfs_inode_t *at)

Variables

- vfs_fsinfo_t **fat32**

5.30.1 Detailed Description

Implementation of FAT32 related functions.

```
The functions in this file are all needed by VFS requirements. The r/w of
FAT32 are PIO 28 which need to be improved for higher speed.
```

5.30.2 Variable Documentation

5.30.2.1 fat32

```
vfs_fsinfo_t fat32
```

Initial value:

```
= {
    .name = "fat32",
    .istemp = false,
    .filelist = {0},
    .open = fat32_open,
    .mount = fat32_mount,
    .mknode = fat32_mknode,
    .sync = fat32_sync,
    .refresh = fat32_refresh,
    .read = fat32_read,
    .getdent = fat32_getdent,
    .write = fat32_write,
}
```

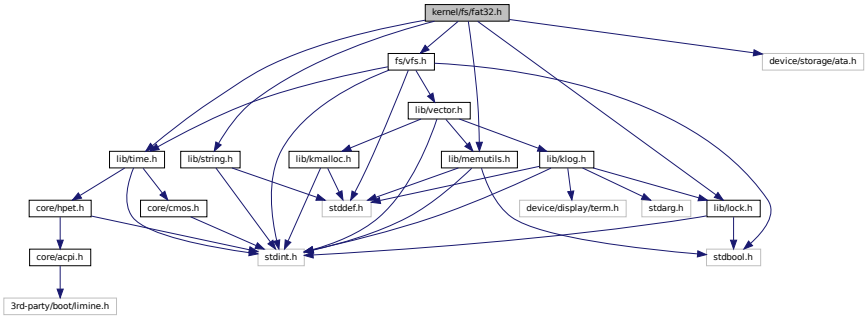
5.31 kernel/fs/fat32.h File Reference

Definition of FAT32 related data structures and functions.

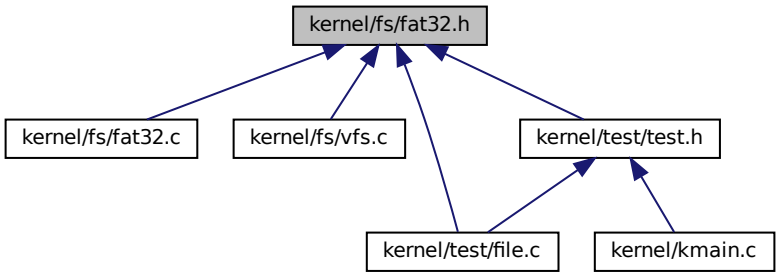
```
#include <fs/vfs.h>
#include <lib/memutils.h>
#include <lib/string.h>
#include <lib/lock.h>
#include <lib/time.h>
```

```
#include <device/storage/ata.h>
```

Include dependency graph for fat32.h:



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



Data Structures

- struct fat_extbs_32_t
- struct fat_extbs_16_t
- struct fat_bs_t
- struct fat_dir_entry_t
- struct fat_lfn_entry_t
- struct fat32_bs_info_t
- struct fat32_entry_t
- struct fat32_ident_t
- struct fat32_ident_item_t

Macros

- #define **FAT32_ATTR_READ_ONLY** 0x01
- #define **FAT32_ATTR_HIDDEN** 0x02
- #define **FAT32_ATTR_SYSTEM** 0x04
- #define **FAT32_ATTR_VOLUME_ID** 0x08
- #define **FAT32_ATTR_DIRECTORY** 0x10
- #define **FAT32_ATTR_ARCHIVE** 0x20
- #define **FAT32_ATTR_LONGNAME** 0x0F

Functions

- vfs_inode_t * **fat32_mount** (vfs_inode_t *at)
- vfs_tnode_t * **fat32_open** (vfs_inode_t *this, const char *path)
- int64_t **fat32_mknode** (vfs_tnode_t *this)
- int64_t **fat32_read** (vfs_inode_t *this, size_t offset, size_t len, void *buff)
- int64_t **fat32_write** (vfs_inode_t *this, size_t offset, size_t len, const void *buff)
- int64_t **fat32_sync** (vfs_inode_t *this)
- int64_t **fat32_refresh** (vfs_inode_t *this)
- int64_t **fat32_getdent** (vfs_inode_t *this, size_t pos, vfs_dirent_t *dirent)

Variables

- vfs_fsinfo_t **fat32**

5.31.1 Detailed Description

Definition of FAT32 related data structures and functions.

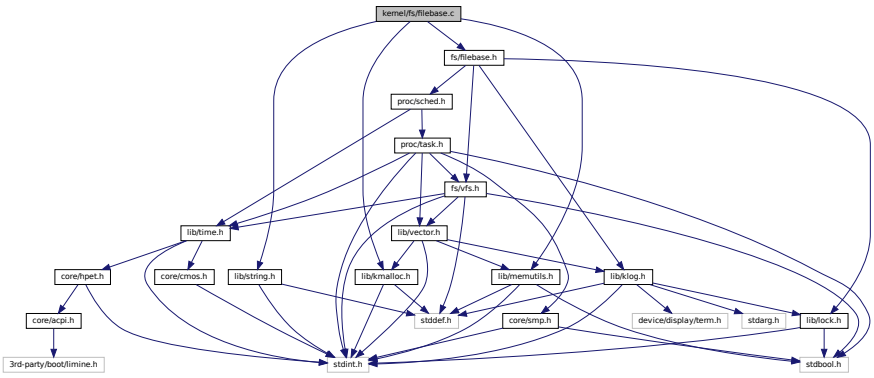
```
FAT 32 was introduced to us by Windows95-B and Windows98. FAT32 solved some of
FAT's problems. No more 64K max clusters! Although FAT32 uses 32 bits per FAT
entry, only the bottom 28 bits are actually used to address clusters on the
disk (top 4 bits are reserved). With 28 bits per FAT entry, the filesystem can
address a maximum of about 270 million clusters in a partition. This enables
very large hard disks to still maintain reasonably small cluster sizes and
thus reduce slack space between files.
```

5.32 kernel/fs/filebase.c File Reference

Implementation of fundamental VFS file node related functions.

```
#include <fs/filebase.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <lib/string.h>
```

Include dependency graph for filebase.c:



Functions

- **vec_extern** (vfs_node_desc_t *, vfs_openfiles)
- vfs_tnode_t * **vfs_alloc_tnode** (const char *name, vfs_inode_t *inode, vfs_inode_t *parent)
- vfs_inode_t * **vfs_alloc_inode** (vfs_node_type_t type, uint32_t perms, uint32_t uid, vfs_fsinfo_t *fs, vfs_tnode_t *mountpoint)
- void **vfs_free_nodes** (vfs_tnode_t *tnode)
- vfs_node_desc_t * **vfs_handle_to_fd** (vfs_handle_t handle)
- vfs_tnode_t * **vfs_path_to_node** (const char *path, uint8_t mode, vfs_node_type_t create_type)

5.32.1 Detailed Description

Implementation of fundamental VFS file node related functions.

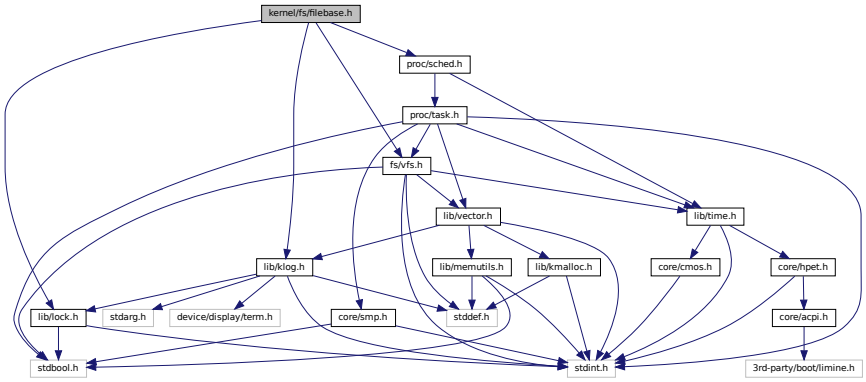
```
This file contains implementation of all inode (index node) and tnode (tree
node) related functions, e.g., alloc, free and node to fd (file descriptor),
path to node conversions.
```

5.33 kernel/fs/filebase.h File Reference

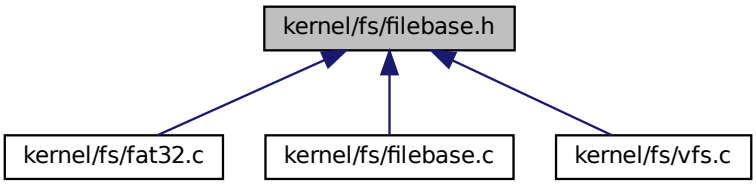
Definition of fundamental VFS file node related functions.

```
#include <lib/lock.h>
#include <lib/klog.h>
#include <proc/sched.h>
#include <fs/vfs.h>
```

Include dependency graph for filebase.h:



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



Macros

- #define **IS_TRAVERSABLE**(x) ((x)->type == VFS_NODE_FOLDER || (x)->type == VFS_NODE_MOUN↩ TPOINT)
- #define **NO_CREATE** 0b0001U
- #define **CREATE** 0b0010U
- #define **ERR_ON_EXIST** 0b0100U

Functions

- vfs_tnode_t * **vfs_alloc_tnode** (const char *name, vfs_inode_t *inode, vfs_inode_t *parent)

- vfs_inode_t * **vfs_alloc_inode** (vfs_node_type_t type, uint32_t perms, uint32_t uid, vfs_fsinfo_t *fs, vfs_tnode_t *mnt)
- void **vfs_free_nodes** (vfs_tnode_t *tnode)
- vfs_node_desc_t * **vfs_handle_to_fd** (vfs_handle_t handle)
- vfs_tnode_t * **vfs_path_to_node** (const char *path, uint8_t mode, vfs_node_type_t create_type)

Variables

- lock_t **vfs_lock**
- vfs_tnode_t **vfs_root**

5.33.1 Detailed Description

Definition of fundamental VFS file node related functions.

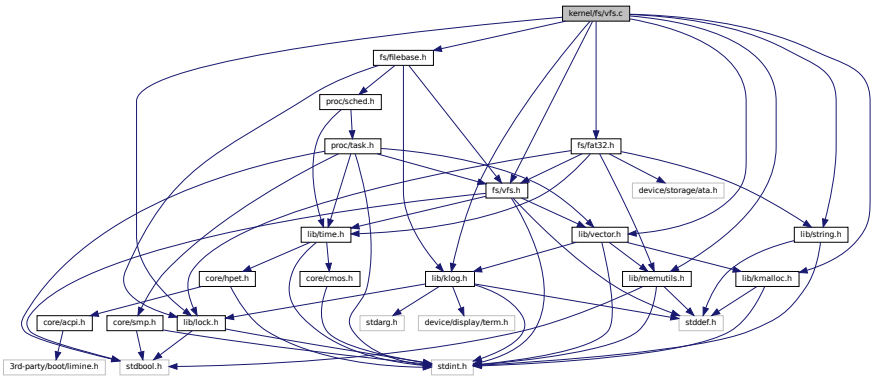
```
This file contains definition of inode (index node) and tnode (tree node)
related functions, e.g., alloc, free and node to fd (file descriptor), path
to node conversions.
```

5.34 kernel/fs/vfs.c File Reference

Implementation of VFS related functions.

```
#include <fs/vfs.h>
#include <fs/filebase.h>
#include <fs/fat32.h>
#include <lib/klog.h>
#include <lib/kmalloc.h>
#include <lib/lock.h>
#include <lib/string.h>
#include <lib/memutils.h>
#include <lib/vector.h>
```

Include dependency graph for vfs.c:



Functions

- **vec_new_static** (vfs_fsinfo_t *, vfs_fslist)
- **vec_new** (vfs_node_desc_t *, vfs_openfiles)
- void **vfs_debug** ()
- void **vfs_register_fs** (vfs_fsinfo_t *fs)
- vfs_fsinfo_t * **vfs_get_fs** (char *name)
- void **vfs_init** ()
- int64_t **vfs_create** (char *path, vfs_node_type_t type)

- int64_t **vfs_chmod** (vfs_handle_t handle, int32_t newperms)
- int64_t **vfs_mount** (char *device, char *path, char *fsname)
- int64_t **vfs_read** (vfs_handle_t handle, size_t len, void *buff)
- int64_t **vfs_write** (vfs_handle_t handle, size_t len, const void *buff)
- int64_t **vfs_seek** (vfs_handle_t handle, size_t pos)
- void **vfs_get_parent_dir** (const char *path, char *parent)
- vfs_handle_t **vfs_open** (char *path, vfs_openmode_t mode)
- int64_t **vfs_close** (vfs_handle_t handle)
- int64_t **vfs_refresh** (vfs_handle_t handle)
- int64_t **vfs_getdent** (vfs_handle_t handle, vfs_dirent_t *dirent)

Variables

- lock_t **vfs_lock**
- vfs_tnode_t **vfs_root**

5.34.1 Detailed Description

Implementation of VFS related functions.

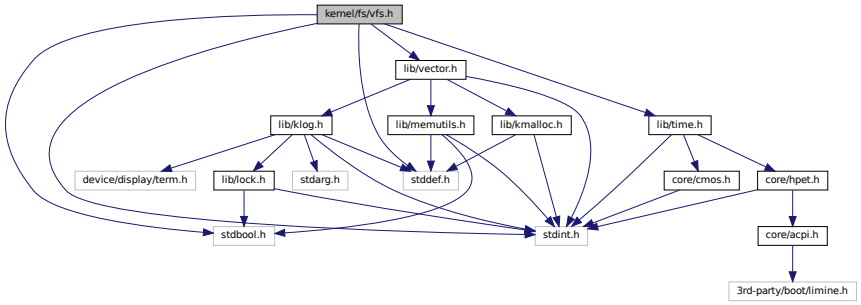
```
Main VFS functions are implemented in this file.
```

5.35 kernel/fs/vfs.h File Reference

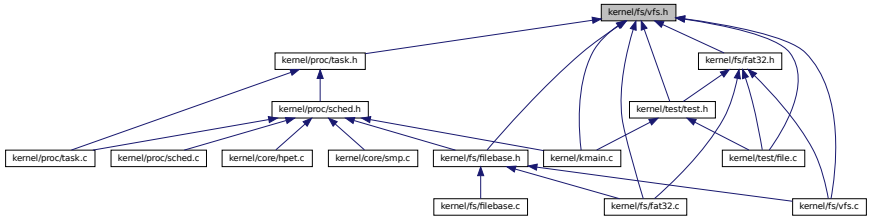
Definition of VFS related data structures and functions.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <lib/vector.h>
#include <lib/time.h>
```

Include dependency graph for vfs.h:



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



Data Structures

- struct vfs_dirent_t
- struct vfs_fsinfo_t
- struct vfs_tnode_t
- struct vfs_inode_t
- struct vfs_node_desc_t

Macros

- #define **VFS_MAX_PATH_LEN** 4096
- #define **VFS_MAX_NAME_LEN** 256
- #define **VFS_INVALID_HANDLE** -1

Typedefs

- typedef int **vfs_handle_t**
- typedef struct vfs_inode_t **vfs_inode_t**
- typedef struct vfs_tnode_t **vfs_tnode_t**
- typedef struct vfs_fsinfo_t **vfs_fsinfo_t**

Enumerations

- enum **vfs_node_type_t** { **VFS_NODE_FILE**, **VFS_NODE_FOLDER**, **VFS_NODE_BLOCK_DEVICE**, **VFS_NODE_CHAR_DEVICE**, **VFS_NODE_MOUNTPOINT** }
- enum **vfs_openmode_t** { **VFS_MODE_READ**, **VFS_MODE_WRITE**, **VFS_MODE_READWRITE** }

Functions

- void **vfs_init** ()
- void **vfs_register_fs** (vfs_fsinfo_t *fs)
- vfs_fsinfo_t * **vfs_get_fs** (char *name)
- void **vfs_debug** ()
- vfs_handle_t **vfs_open** (char *path, vfs_openmode_t mode)
- int64_t **vfs_create** (char *path, vfs_node_type_t type)
- int64_t **vfs_close** (vfs_handle_t handle)
- int64_t **vfs_seek** (vfs_handle_t handle, size_t pos)
- int64_t **vfs_read** (vfs_handle_t handle, size_t len, void *buff)
- int64_t **vfs_write** (vfs_handle_t handle, size_t len, const void *buff)
- int64_t **vfs_chmod** (vfs_handle_t handle, int32_t newperms)
- int64_t **vfs_refresh** (vfs_handle_t handle)
- int64_t **vfs_getdent** (vfs_handle_t handle, vfs_dirent_t *dirent)
- int64_t **vfs_mount** (char *device, char *path, char *fsname)

5.35.1 Detailed Description

Definition of VFS related data structures and functions.

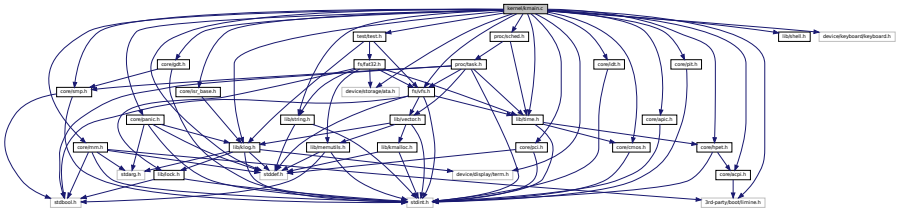
```
All VFS data structures and interfaces are defined in this file.
```

5.36 kernel/kmain.c File Reference

Entry function of HanOS kernel.

```
#include <stddef.h>
#include <3rd-party/boot/limine.h>
#include <lib/time.h>
#include <lib/klog.h>
#include <lib/string.h>
#include <lib/shell.h>
#include <core/mm.h>
#include <core/gdt.h>
#include <core/idt.h>
#include <core/isr_base.h>
#include <core/smp.h>
#include <core/cmos.h>
#include <core/acpi.h>
#include <core/apic.h>
#include <core/hpet.h>
#include <core/panic.h>
#include <core/pci.h>
#include <core/pit.h>
#include <device/display/term.h>
#include <device/keyboard/keyboard.h>
#include <device/storage/ata.h>
#include <proc/sched.h>
#include <fs/vfs.h>
#include <test/test.h>
```

Include dependency graph for kmain.c:



Enumerations

- enum { **CURSOR_INVISIBLE** = 0, **CURSOR_VISIBLE**, **CURSOR_HIDE** }

Functions

- _Noreturn void **kcursor** (task_id_t tid)
- _Noreturn void **kshell** (task_id_t tid)
- void **kmain** (void)

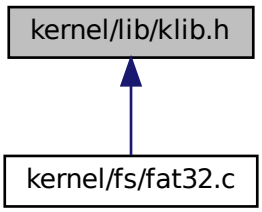
5.36.1 Detailed Description

Entry function of HanOS kernel.

```
Lots of system initializations will be processed here.

History:
  Feb 19, 2022  Added CLI task which supports some simple commands.
  May 21, 2022  Changed boot protocol to limine with corresponding
                modifications.
```

5.37 kernel/lib/klib.h File Reference

Definition of fundamental data structures, macros and functions for the kernel.
This graph shows which files directly or indirectly include this file:



Macros

- #define **MIN**(x, y) ((x) < (y) ? (x) : (y))
- #define **MAX**(x, y) ((x) > (y) ? (x) : (y))
- #define **DIV_ROUNDUP**(a, b) (((a) + ((b) - 1)) / (b))

5.37.1 Detailed Description

Definition of fundamental data structures, macros and functions for the kernel.

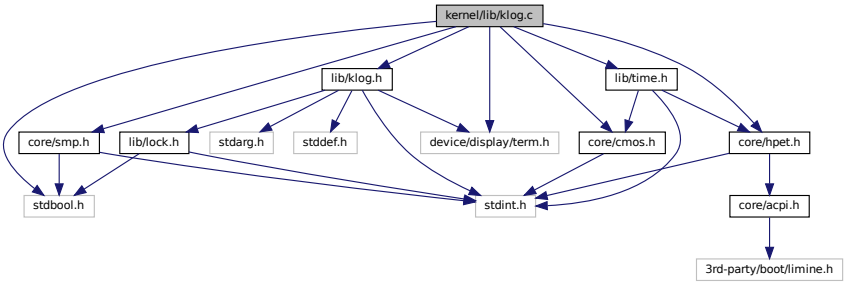
```
Currently this file includes MIN, MAX, DIV_ROUNDUP macros.
```

5.38 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 <core/smp.h>
```

Include dependency graph for klog.c:



Functions

- void **klog_refresh** (int mode)
- void **klog_init** ()
- void **klog_vprintf_core** (int mode, const char *s, va_list args)
- void **klog_vprintf_wrapper** (int mode, const char *s,...)
- void **klog_lock** (void)
- void **klog_unlock** (void)
- void **klog_vprintf** (klog_level_t level, const char *s,...)
- void **kprintf** (const char *s,...)

5.38.1 Detailed Description

Implementation of kernel log related functions.

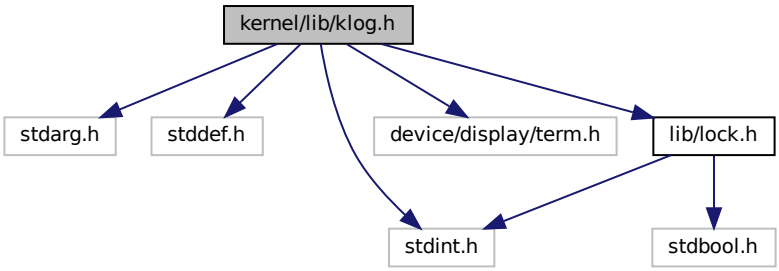
```
A kernel-level log system was implemented. As the first step, it
mainly supports information display.
```

5.39 kernel/lib/klog.h File Reference

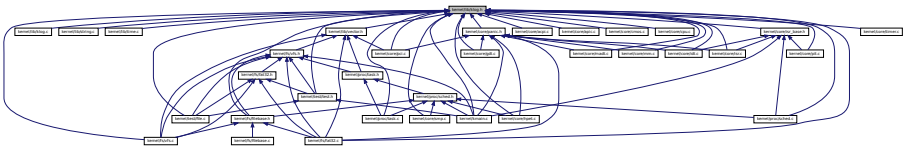
Definition of kernel log related functions.

```
#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_vprintf(KLOG_LEVEL_VERBOSE, s, ##__VA_ARGS__)
- #define **klogd**(s, ...) klog_vprintf(KLOG_LEVEL_DEBUG, s, ##__VA_ARGS__)
- #define **klogi**(s, ...) klog_vprintf(KLOG_LEVEL_INFO, s, ##__VA_ARGS__)
- #define **klogw**(s, ...) klog_vprintf(KLOG_LEVEL_WARN, s, ##__VA_ARGS__)
- #define **kloge**(s, ...) klog_vprintf(KLOG_LEVEL_ERROR, s, ##__VA_ARGS__)
- #define **klogu**(s, ...) klog_vprintf(KLOG_LEVEL_UNK, s, ##__VA_ARGS__)
- #define **klog_printf**(s, ...) klog_vprintf(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)
- void **klog_vprintf** (klog_level_t level, const char *,...)
- void **klog_lock** (void)
- void **klog_unlock** (void)
- void **klog_refresh** (int mode)
- void **kprintf** (const char *,...)

5.39.1 Detailed Description

Definition of kernel log related functions.

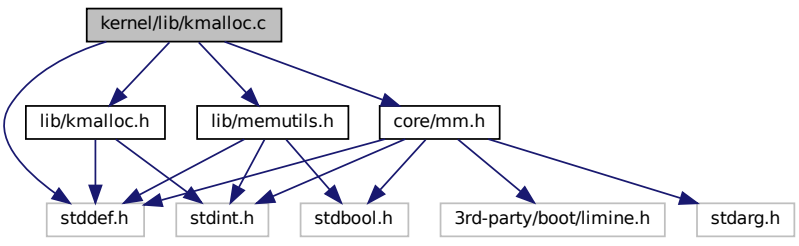
```
A kernel-level log system was implemented. As the first step, it
mainly supports information display.
```

5.40 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)
- void ∗ **umalloc** (uint64_t size)
- void **umfree** (void ∗addr)

5.40.1 Detailed Description

Implementation of memory allocation related functions.

```
e.g., malloc, free and realloc.
```

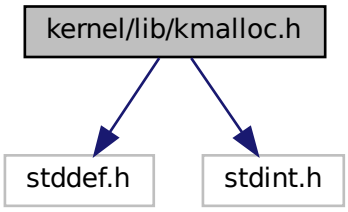
Todo Memory allocation should be improved for better efficiency.

5.41 kernel/lib/kmalloc.h File Reference

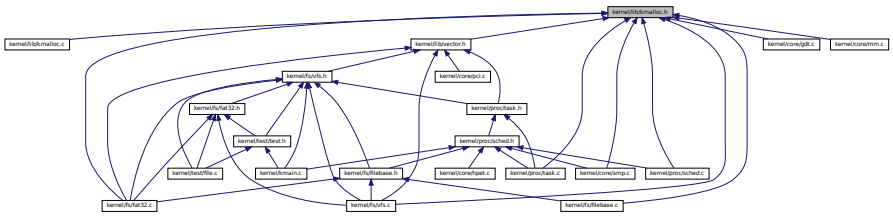
Definition of memory allocation related functions.

```
#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)
- void ∗ **umalloc** (uint64_t size)
- void **umfree** (void ∗addr)

5.41.1 Detailed Description

Definition of memory allocation related functions.

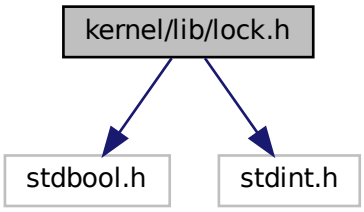
```
e.g., malloc, free and realloc.
```

5.42 kernel/lib/lock.h File Reference

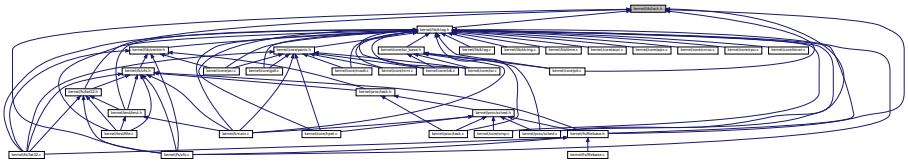
Definition of lock related data structures and functions.

```
#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)

5.42.1 Detailed Description

Definition of lock related data structures and functions.

```
e.g., lock new, lock and release.
```

5.42.2 Macro Definition Documentation

5.42.2.1 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");                                          \
    }
```

5.42.2.2 lock_release

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

Value:

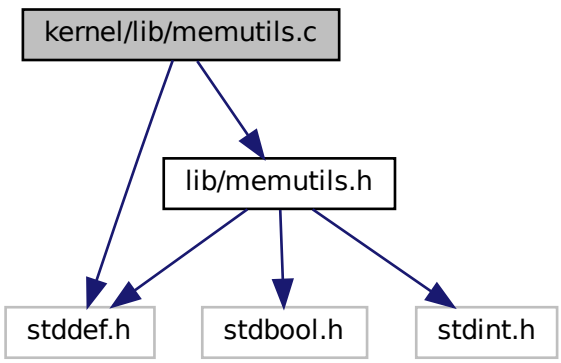
```
    {                                                   \
        asm volatile("push %[flags];"                   \
                     "lock btrl $0, %[lock];"           \
                     "popfq;"                           \
                     : [lock] "=m"((s)->lock)           \
      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

5.43 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 ∗dst, 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)

5.43.1 Detailed Description

Implementation of memory operation related functions.

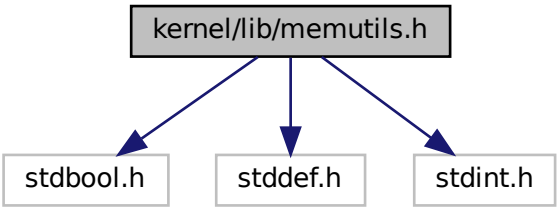
```
e.g., comparison, set value and copy.
```

5.44 kernel/lib/memutils.h File Reference

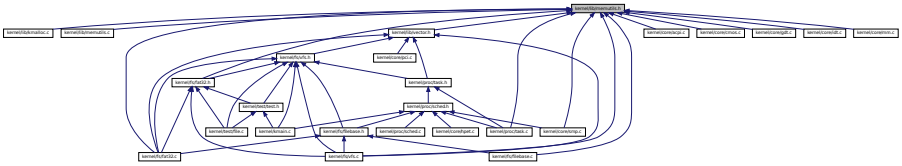
Definition of memory operation related functions.

```
#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 ∗dst, const void ∗src, uint64_t len)

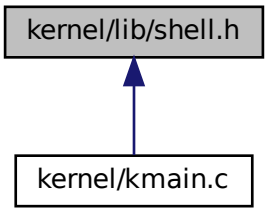
5.44.1 Detailed Description

Definition of memory operation related functions.

```
e.g., comparison, set value and copy.
```

5.45 kernel/lib/shell.h File Reference

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



Data Structures

- struct command_t

Typedefs

- typedef void(∗ **command_proc_t**) (void)

5.45.1 Detailed Description

Definition of shell related data structures and functions.

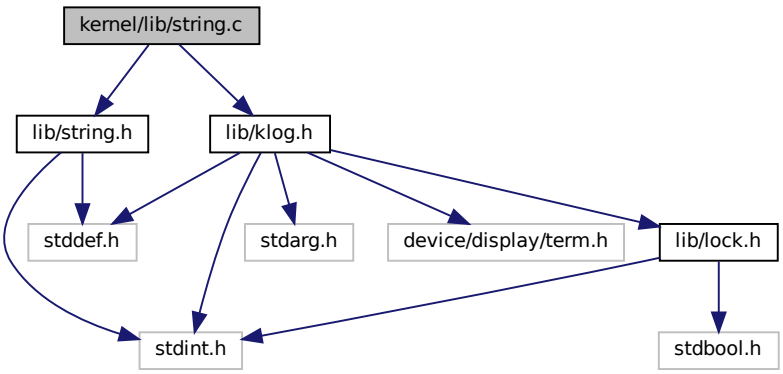
```
- The command and the corresponding processing function.
- desc[] stores information to be displayed in the help command.
```

5.46 kernel/lib/string.c File Reference

Implementation of string operation related functions.

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

Include dependency graph for string.c:



Functions

- int **strlen** (const char ∗s)
- int **strcmp** (const char ∗a, const char ∗b)
- int **strncmp** (const char ∗a, const char ∗b, size_t len)
- int **strcpy** (char ∗dest, const char ∗src)
- int **strcat** (char ∗dest, const char ∗src)
- char ∗ **strchrnul** (const char ∗s, int c)
- int **islower** (int c)
- int **tolower** (int c)
- int **toupper** (int c)

5.46.1 Detailed Description

Implementation of string operation related functions.

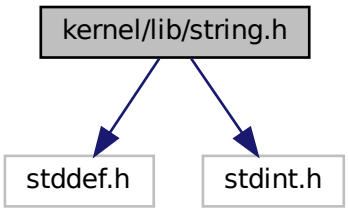
```
String functions including length, comparison, copy etc.
```

5.47 kernel/lib/string.h File Reference

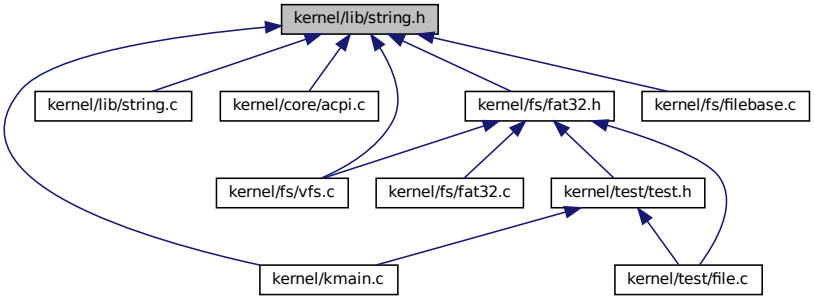
Definition of string operation related functions.

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

Include dependency graph for string.h:



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



Functions

- int **strlen** (const char *s)
- int **strcmp** (const char *a, const char *b)
- int **strncmp** (const char *a, const char *b, size_t len)
- int **strcpy** (char *dest, const char *src)
- int **strcat** (char *dest, const char *src)
- char * **strchrnul** (const char *s, int c)
- int **islower** (int)
- int **tolower** (int)
- int **toupper** (int)

5.47.1 Detailed Description

Definition of string operation related functions.

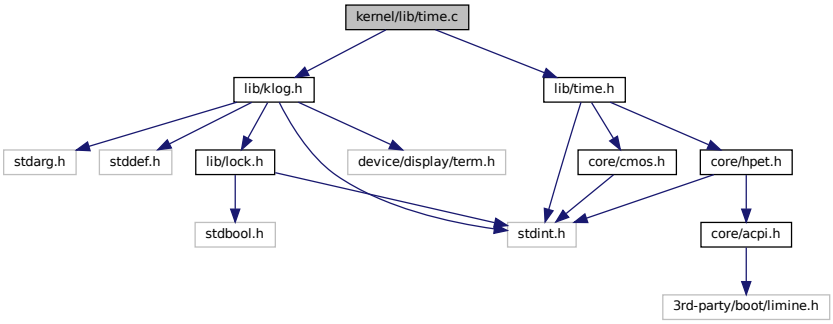
```
String functions including length, comparison, copy etc.
```

5.48 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)

5.48.1 Detailed Description

Implementation of time related functions.

```
e.g., localtime...
```

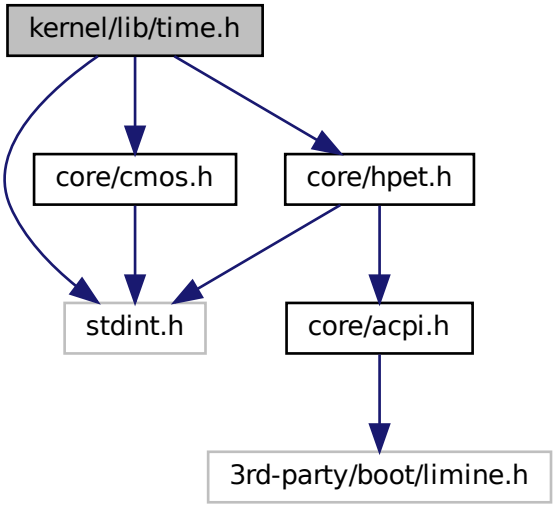
5.49 kernel/lib/time.h File Reference

Definition of time related data structures and functions.

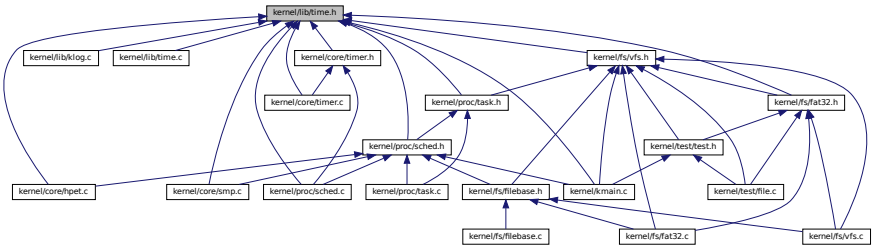
```
#include <stdint.h>
#include <core/hpet.h>
```

```
#include <core/cmos.h>
```

Include dependency graph for time.h:



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



Data Structures

- 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)

5.49.1 Detailed Description

Definition of time related data structures and functions.

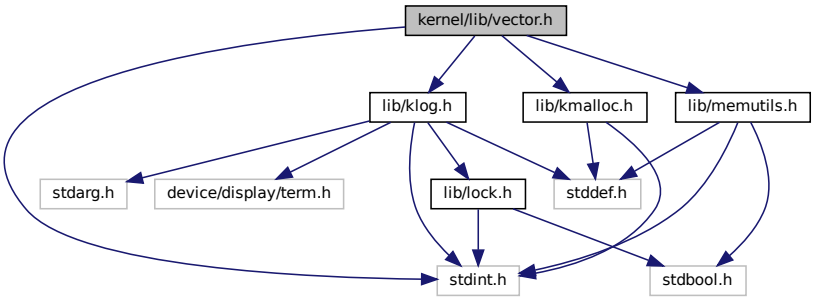
```
e.g., time value, time zone and time information.
```

5.50 kernel/lib/vector.h File Reference

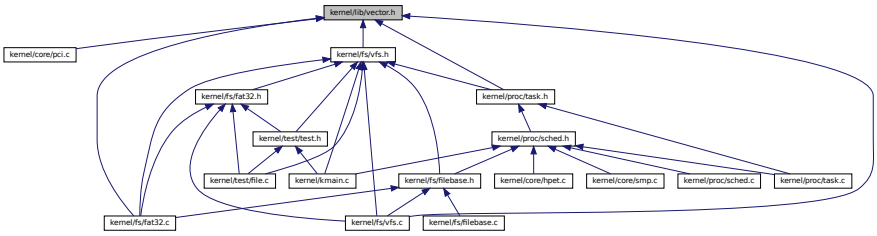
Definition of vector related data structures and functions.

```
#include <stdint.h>
#include <lib/klog.h>
#include <lib/memutils.h>
#include <lib/kmalloc.h>
```

Include dependency graph for vector.h:



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



Macros

- #define **VECTOR_RESIZE_FACTOR** 4
- #define **vec_struct**(type)
- #define **vec_extern**(type, name) extern vec_struct(type) name
- #define **vec_new**(type, name) vec_struct(type) name = {0}
- #define **vec_new_static**(type, name) static vec_new(type, name)
- #define **vec_push_back**(vec, elem)
- #define **vec_length**(vec) (vec)->len
- #define **vec_at**(vec, index) (vec)->data[index]
- #define **vec_erase**(vec, index)
- #define **vec_erase_val**(vec, val)

5.50.1 Detailed Description

Definition of vector related data structures and functions.

```
This file provides the implementation of a variable length array. All the
functions are defined in macros.
```

5.50.2 Macro Definition Documentation

5.50.2.1 vec_erase

```
#define vec_erase(
                vec,
                index )
```

Value:

```
    {                                                                   \
        memcpy(&((vec)->data[index]), &((vec)->data[index + 1]),        \
               sizeof((vec)->data[0]) * (vec)->len - index - 1);        \
        (vec)->len--;                                                   \
    }
```

5.50.2.2 vec_erase_val

```
#define vec_erase_val(
                vec,
                val )
```

Value:

```
    {                                                                   \
        for(size_t __i = 0; __i < (vec)->len; __i++) {                  \
            if (vec_at(vec, __i) == (val)) {                            \
                vec_erase(vec, __i);                                    \
                break;                                                  \
            }                                                           \
        }                                                               \
    }
```

5.50.2.3 vec_push_back

```
#define vec_push_back(
                vec,
                elem )
```

Value:

```
    {                                                                   \
        (vec)->len++;                                                   \
        if ((vec)->capacity < (vec)->len * sizeof(elem)) {              \
            (vec)->capacity = (vec)->len * sizeof(elem)                 \
                              * VECTOR_RESIZE_FACTOR;                   \
            (vec)->data = kmrealloc((vec)->data, (vec)->capacity);      \
        }                                                               \
        (vec)->data[(vec)->len - 1] = elem;                             \
    }
```

5.50.2.4 vec_struct

```
#define vec_struct(
                type )
```

Value:

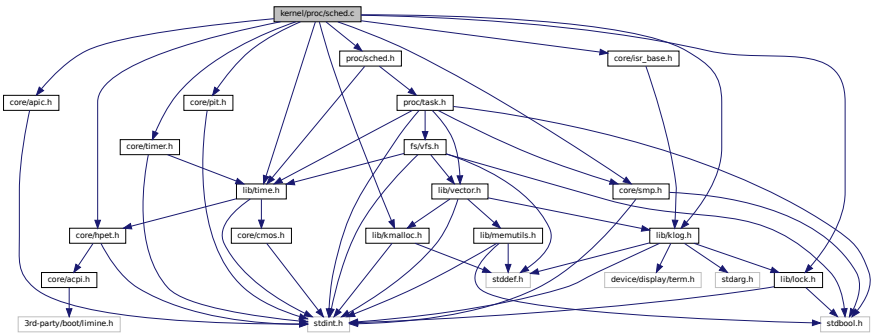
```
    struct {                                                            \
        size_t len;                                                     \
        size_t capacity;                                                \
        type*  data;                                                    \
    }
```

5.51 kernel/proc/sched.c File Reference

Implementation of scheduling related functions.

```
#include <lib/klog.h>
#include <lib/lock.h>
#include <lib/time.h>
#include <lib/kmalloc.h>
#include <proc/sched.h>
#include <core/smp.h>
#include <core/timer.h>
#include <core/apic.h>
#include <core/hpet.h>
#include <core/pit.h>
#include <core/isr_base.h>
```

Include dependency graph for sched.c:



Macros

- #define **TIMESLICE_DEFAULT** MILLIS_TO_NANOS(1)

Functions

- **vec_new_static** (task_t ∗, tasks_active)
- void **enter_context_switch** (void ∗v)
- void **exit_context_switch** (task_t ∗next)
- void **do_context_switch** (void ∗stack)
- void **sched_sleep** (time_t millis)
- task_t ∗ **sched_get_current_task** ()
- uint64_t **sched_get_ticks** ()
- void **sched_init** (uint16_t cpu_id)
- uint16_t **sched_get_cpu_num** ()
- void **sched_add** (void(∗entry)(task_id_t))

5.51.1 Detailed Description

Implementation of scheduling related functions.

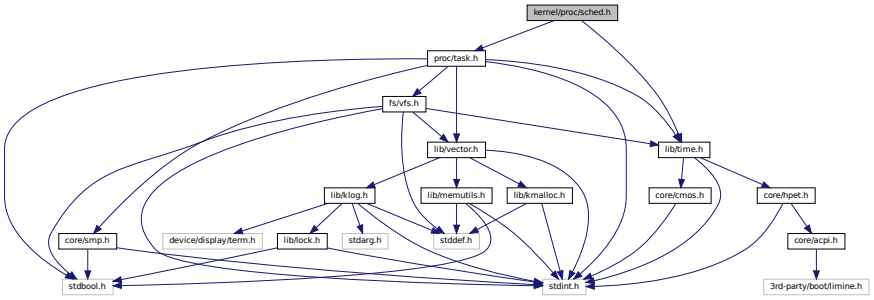
```
Context Switching, Scheduling Algorithms etc.

History:
Apr 20, 2022 - 1. Redesign the task queue based on vector data structue.
               2. Scheduler starts working after all processors are launched
                  to avoid GPF exception.
```

5.52 kernel/proc/sched.h File Reference

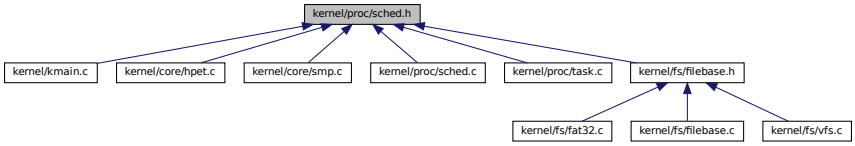
Definition of scheduling related functions.

```
#include <proc/task.h>
#include <lib/time.h>
```

Include dependency graph for sched.h:



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



Functions

- void **sched_init** (uint16_t cpu_id)
- void **sched_add** (void(∗entry)(task_id_t))
- void **sched_sleep** (time_t ms)
- task_t ∗ **sched_get_current_task** ()
- uint16_t **sched_get_cpu_num** ()
- uint64_t **sched_get_ticks** ()

5.52.1 Detailed Description

Definition of scheduling related functions.

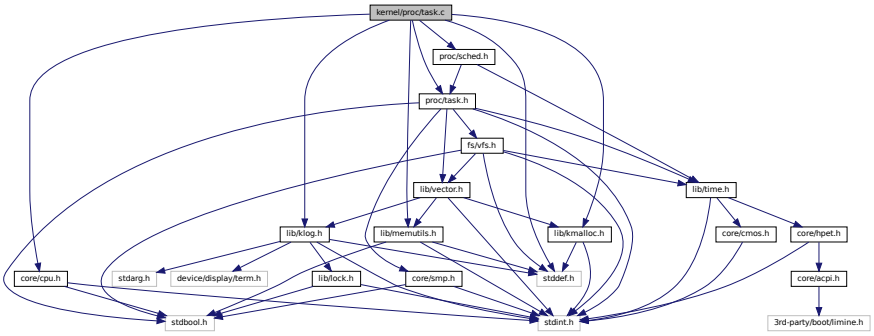
```
Include Context Switching, Scheduling Algorithms etc.
```

5.53 kernel/proc/task.c File Reference

Implementation of task related functions.

```
#include <stddef.h>
#include <proc/task.h>
#include <proc/sched.h>
#include <lib/kmalloc.h>
#include <lib/memutils.h>
#include <lib/klog.h>
```

```
#include <core/cpu.h>
```
Include dependency graph for task.c:



Functions

- task_t ∗ **task_make** (void(∗entry)(task_id_t), task_priority_t priority, task_mode_t mode)

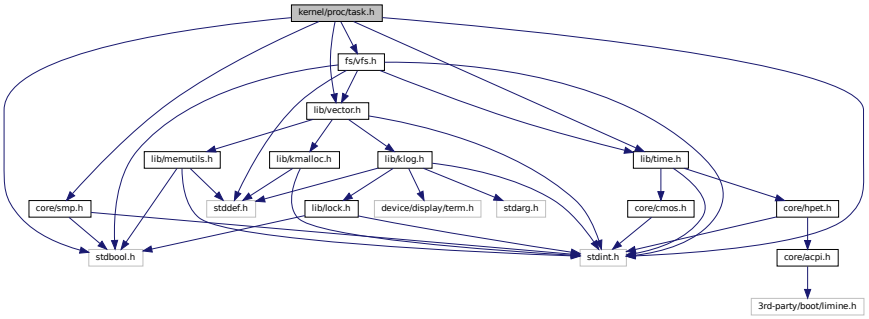
5.53.1 Detailed Description

Implementation of task related functions.

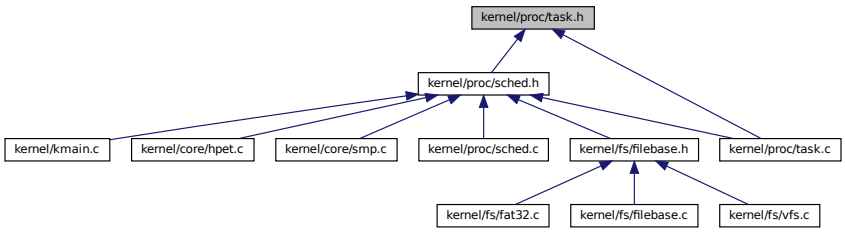
```
Create and return task data structure which contains registers and other task
related information.
```

5.54 kernel/proc/task.h File Reference

Definition of task related functions.
```
#include <stdbool.h>
#include <stdint.h>
#include <lib/time.h>
#include <lib/vector.h>
#include <core/smp.h>
#include <fs/vfs.h>
```
Include dependency graph for task.h:



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



Data Structures

- struct task_regs_t
- struct task_t

Macros

- #define **KSTACK_SIZE** 4096
- #define **DEFAULT_KMODE_CS** 0x08
- #define **DEFAULT_KMODE_SS** 0x10
- #define **DEFAULT_UMODE_CS** 0x1b
- #define **DEFAULT_UMODE_SS** 0x23
- #define **DEFAULT_RFLAGS** 0x0202
- #define **TID_MAX** UINT16_MAX

Typedefs

- typedef uint16_t **task_id_t**
- typedef uint8_t **task_priority_t**
- typedef struct task_t **task_t**

Enumerations

- enum **task_mode_t** { **TASK_KERNEL_MODE**, **TASK_USER_MODE** }
- enum **task_status_t** { **TASK_READY**, **TASK_RUNNING**, **TASK_SLEEPING**, **TASK_DEAD** }

Functions

- task_t ∗ **task_make** (void(∗entry)(task_id_t), task_priority_t priority, task_mode_t mode)

5.54.1 Detailed Description

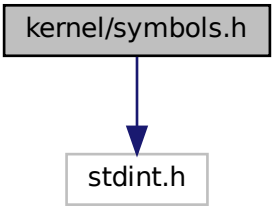
Definition of task related functions.

```
Create and return task data structure which contains registers and other task
related information.
```

5.55 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** [ ]

5.55.1 Detailed Description

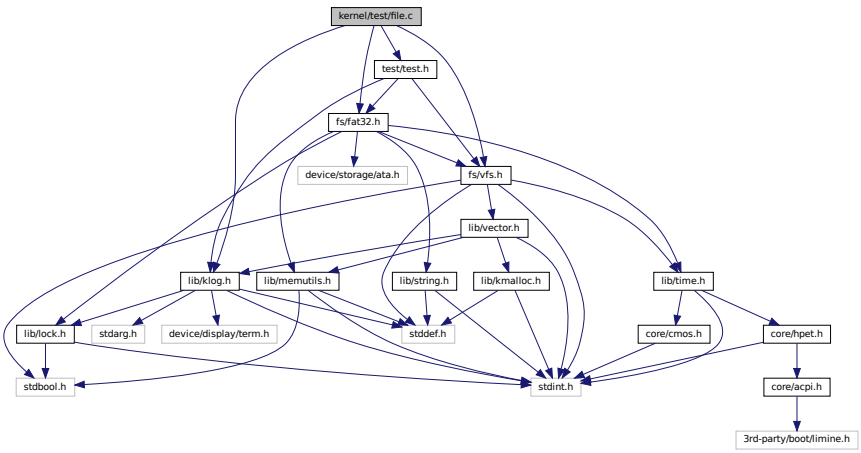
Definition of symbol related data structures.

```
Symbols are used for backtrace when kernel is crashed. It can help to
provide context information for debugging.
```

5.56 kernel/test/file.c File Reference

Implementation of file test functions.

```
#include <fs/vfs.h>
#include <fs/fat32.h>
#include <lib/klog.h>
#include <test/test.h>
```
Include dependency graph for file.c:



Functions

- void **dir_test** (void)
- void **file_test** (void)

5.56.1 Detailed Description

Implementation of file test functions.

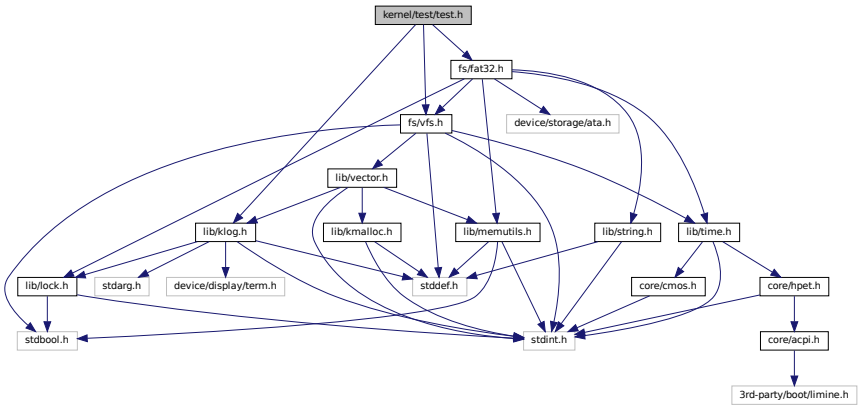
```
The file test functions in this file can be called in kmain() file..
```

5.57 kernel/test/test.h File Reference

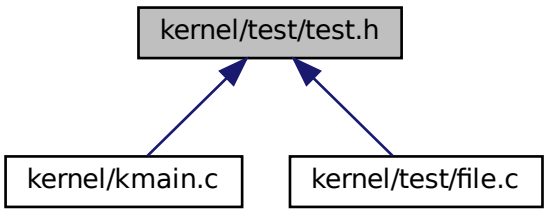
Definition of some test functions.

```
#include <fs/vfs.h>
#include <fs/fat32.h>
#include <lib/klog.h>
```

Include dependency graph for test.h:



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



Functions

- void **file_test** (void)
- void **dir_test** (void)

5.57.1 Detailed Description

Definition of some test functions.

```
Test functions are mainly designed for the command line interface.
```

Index