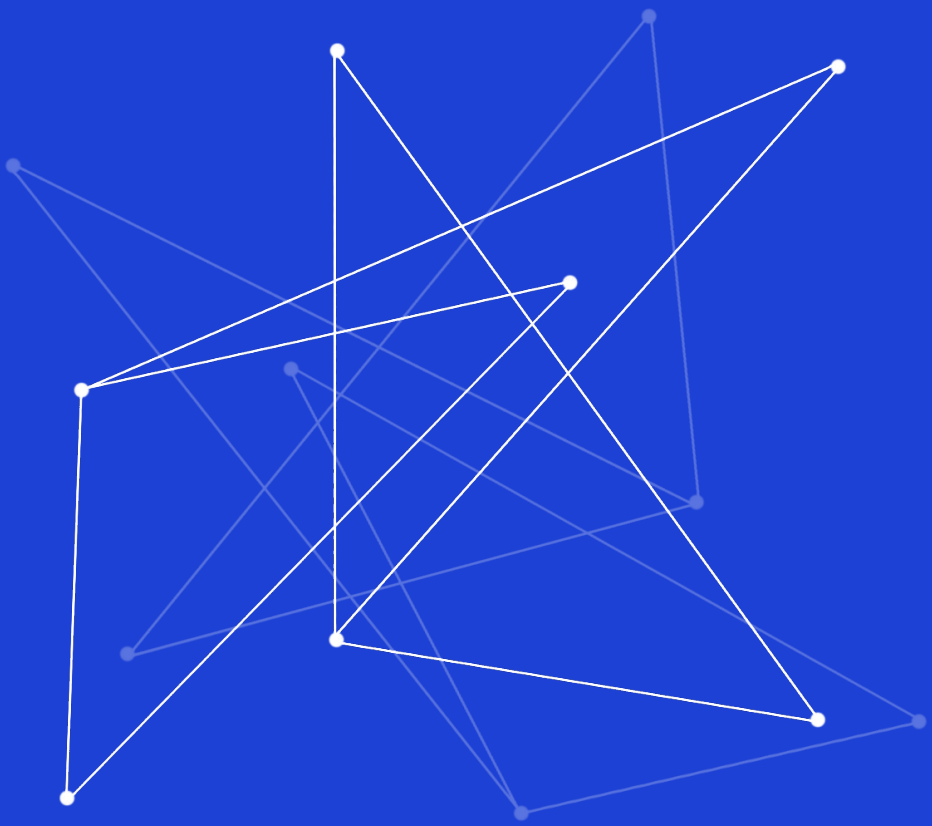
THE NULL OPERATING SYSTEM SPECIFICATION



nos v0.0.2 2024 (@alkuzin)

NOS - The Null Operating System

0.0.2

Generated by Doxygen 1.9.8

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 gdt_entry_s Struct Reference

GDT entry (segment descriptor) - tells the CPU the attributes of a given segment.

`#include <gdt.h>`

Data Fields

- u16 limit
- u16 base_low
- u8 base_mid
- u8 access
- u8 flags
- u8 base_high

3.1.1 Detailed Description

GDT entry (segment descriptor) - tells the CPU the attributes of a given segment.

Warning

Order of segment descriptor content is important!

3.1.2 Field Documentation

3.1.2.1 access

`u8 gdt_entry_s::access`

3.1.2.2 base_high

`u8 gdt_entry_s::base_high`

3.1.2.3 base_low

```
u16 gdt_entry_s::base_low
```

3.1.2.4 base_mid

```
u8 gdt_entry_s::base_mid
```

3.1.2.5 flags

```
u8 gdt_entry_s::flags
```

3.1.2.6 limit

```
u16 gdt_entry_s::limit
```

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

- gdt.h

3.2 gdt_ptr_s Struct Reference

GDT pointer structure.

```
#include <gdt.h>
```

Data Fields

- u16 limit
- u32 base

3.2.1 Detailed Description

GDT pointer structure.

3.2.2 Field Documentation

3.2.2.1 base

```
u32 gdt_ptr_s::base
```

3.2.2.2 limit

```
u16 gdt_ptr_s::limit
```

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

- gdt.h

3.3 idt_entry_s Struct Reference

Interrupt Descriptor Table entry structure.

```
#include <idt.h>
```

Data Fields

- u16 base_low
- u16 sel
- u8 always0
- u8 flags
- u16 base_high

3.3.1 Detailed Description

Interrupt Descriptor Table entry structure.

3.3.2 Field Documentation

3.3.2.1 always0

```
u8 idt_entry_s::always0
```

3.3.2.2 base_high

```
u16 idt_entry_s::base_high
```

3.3.2.3 base_low

```
u16 idt_entry_s::base_low
```

3.3.2.4 flags

```
u8 idt_entry_s::flags
```

3.3.2.5 sel

```
u16 idt_entry_s::sel
```

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

- idt.h

3.4 idt_ptr_s Struct Reference

Interrupt Descriptor Table pointer structure.

```
#include <idt.h>
```

Data Fields

- u16 limit
- u32 base

3.4.1 Detailed Description

Interrupt Descriptor Table pointer structure.

3.4.2 Field Documentation

3.4.2.1 base

```
u32 idt_ptr_s::base
```

3.4.2.2 limit

```
u16 idt_ptr_s::limit
```

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

- idt.h

3.5 int_reg_s Struct Reference

Structure representing interrupt register state.

```
#include <irq.h>
```

Data Fields

- u32 cr2
  Control Register 2.
- u32 ds
  Data Segment.
- u32 edi
  Destination Index.
- u32 esi
  Source Index.
- u32 ebp
  Base Pointer.
- u32 esp
  Stack Pointer.
- u32 ebx
  Base Register.
- u32 edx
  Data Register.
- u32 ecx
  Counter Register.
- u32 eax
  Accumulator Register.
- u32 int_no
  Interrupt Number.
- u32 err_code
  Error Code.
- u32 eip
  Instruction Pointer.
- u32 csm
  Code Segment.
- u32 eflags
  Flags Register.
- u32 useresp
  User Stack Pointer.
- u32 ss
  Stack Segment.

3.5.1 Detailed Description

Structure representing interrupt register state.

3.5.2 Field Documentation

3.5.2.1 cr2

```
u32 int_reg_s::cr2
```

Control Register 2.

3.5.2.2 csm

u32 int_reg_s::csm

Code Segment.

3.5.2.3 ds

u32 int_reg_s::ds

Data Segment.

3.5.2.4 eax

u32 int_reg_s::eax

Accumulator Register.

3.5.2.5 ebp

u32 int_reg_s::ebp

Base Pointer.

3.5.2.6 ebx

u32 int_reg_s::ebx

Base Register.

3.5.2.7 ecx

u32 int_reg_s::ecx

Counter Register.

3.5.2.8 edi

u32 int_reg_s::edi

Destination Index.

3.5.2.9 edx

u32 int_reg_s::edx

Data Register.

3.5.2.10 eflags

```
u32 int_reg_s::eflags
```

Flags Register.

3.5.2.11 eip

```
u32 int_reg_s::eip
```

Instruction Pointer.

3.5.2.12 err_code

```
u32 int_reg_s::err_code
```

Error Code.

3.5.2.13 esi

```
u32 int_reg_s::esi
```

Source Index.

3.5.2.14 esp

```
u32 int_reg_s::esp
```

Stack Pointer.

3.5.2.15 int_no

```
u32 int_reg_s::int_no
```

Interrupt Number.

3.5.2.16 ss

```
u32 int_reg_s::ss
```

Stack Segment.

3.5.2.17 useresp

```
u32 int_reg_s::useresp
```

User Stack Pointer.

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

- irq.h

3.6 kmalloc_block_s Struct Reference

Structure representing a block of memory for kernel dynamic memory allocation.

```
#include <kmalloc.h>
```

Data Fields

- usize size
  size of memory block
- bool is_free
  flag indicating if the block of memory is free
- struct kmalloc_block_s * next
  pointer to the next block of memory

3.6.1 Detailed Description

Structure representing a block of memory for kernel dynamic memory allocation.

3.6.2 Field Documentation

3.6.2.1 is_free

```
bool kmalloc_block_s::is_free
```

flag indicating if the block of memory is free

3.6.2.2 next

```
struct kmalloc_block_s* kmalloc_block_s::next
```

pointer to the next block of memory

3.6.2.3 size

```
usize kmalloc_block_s::size
```

size of memory block

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

- kmalloc.h

3.7 multiboot_aout_symbol_table_s Struct Reference

Structure representing the symbol table for a.out format.

```
#include <multiboot.h>
```

Data Fields

- u32 tabsize
  Size of the symbol table.
- u32 strsize
  Size of the string table.
- u32 addr
  Address of the symbol table.
- u32 reserved
  Reserved field.

3.7.1 Detailed Description

Structure representing the symbol table for a.out format.

3.7.2 Field Documentation

3.7.2.1 addr

```
u32 multiboot_aout_symbol_table_s::addr
```

Address of the symbol table.

3.7.2.2 reserved

```
u32 multiboot_aout_symbol_table_s::reserved
```

Reserved field.

3.7.2.3 strsize

```
u32 multiboot_aout_symbol_table_s::strsize
```

Size of the string table.

3.7.2.4 tabsize

```
u32 multiboot_aout_symbol_table_s::tabsize
```

Size of the symbol table.

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

- multiboot.h

3.8 multiboot_elf_section_header_table_s Struct Reference

Structure representing the section header table for ELF format.

```
#include <multiboot.h>
```

Data Fields

- u32 num
  Number of section headers.
- u32 size
  Size of each section header.
- u32 addr
  Address of the section header table.
- u32 shndx
  Section header index.

3.8.1 Detailed Description

Structure representing the section header table for ELF format.

3.8.2 Field Documentation

3.8.2.1 addr

```
u32 multiboot_elf_section_header_table_s::addr
```

Address of the section header table.

3.8.2.2 num

```
u32 multiboot_elf_section_header_table_s::num
```

Number of section headers.

3.8.2.3 shndx

```
u32 multiboot_elf_section_header_table_s::shndx
```

Section header index.

3.8.2.4 size

```
u32 multiboot_elf_section_header_table_s::size
```

Size of each section header.

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

- multiboot.h

3.9 multiboot_mmap_entry_s Struct Reference

Structure representing a memory map entry in multiboot format.

```
#include <multiboot.h>
```

Data Fields

- u32 size
  Size of the memory map entry.
- u32 addr_low
  Lower address of the memory region.
- u32 addr_high
  Higher address of the memory region.
- u32 len_low
  Lower length of the memory region.
- u32 len_high
  Higher length of the memory region.
- u32 type
  Type of memory region.

3.9.1 Detailed Description

Structure representing a memory map entry in multiboot format.

3.9.2 Field Documentation

3.9.2.1 addr_high

```
u32 multiboot_mmap_entry_s::addr_high
```

Higher address of the memory region.

3.9.2.2 addr_low

```
u32 multiboot_mmap_entry_s::addr_low
```

Lower address of the memory region.

3.9.2.3 len_high

```
u32 multiboot_mmap_entry_s::len_high
```

Higher length of the memory region.

3.9.2.4 len_low

```
u32 multiboot_mmap_entry_s::len_low
```

Lower length of the memory region.

3.9.2.5 size

```
u32 multiboot_mmap_entry_s::size
```

Size of the memory map entry.

3.9.2.6 type

```
u32 multiboot_mmap_entry_s::type
```

Type of memory region.

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

- multiboot.h

3.10 multiboot_t Struct Reference

Type representing multiboot information.

```
#include <multiboot.h>
```

Data Fields

- u32 flags

  Flags indicating available information.
- u32 mem_lower

  Lower memory size in KB.
- u32 mem_upper

  Upper memory size in KB.
- u32 boot_device

  Boot device.
- u32 cmdline

  Command line.
- u32 mods_count

  Number of modules.
- u32 mods_addr

  Address of the module list.
- union {
  struct multiboot_aout_symbol_table_s aout_sym
  A.OUT symbol table.
  struct multiboot_elf_section_header_table_s elf_sec
  ELF section header table.
  } u

  Union for symbol table or section header table.
- u32 mmap_length

  Length of memory map.
- u32 mmap_addr

  Address of memory map.
- u32 drives_length

  Length of drive information.
- u32 drives_addr

  Address of drive information.
- u32 config_table

  Configuration table.
- u32 boot_loader_name

  Boot loader name.
- u32 apm_table

  APM table.
- u32 vbe_control_info

  VBE control information.
- u32 vbe_mode_info

  VBE mode information.
- u16 vbe_mode

  VBE mode.
- u16 vbe_interface_seg

  VBE interface segment.
- u16 vbe_interface_off

  VBE interface offset.
- u16 vbe_interface_len

  VBE interface length.

3.10.1 Detailed Description

Type representing multiboot information.

3.10.2 Field Documentation

3.10.2.1 aout_sym

```
struct multiboot_aout_symbol_table_s multiboot_t::aout_sym
```

A.OUT symbol table.

3.10.2.2 apm_table

```
u32 multiboot_t::apm_table
```

APM table.

3.10.2.3 boot_device

```
u32 multiboot_t::boot_device
```

Boot device.

3.10.2.4 boot_loader_name

```
u32 multiboot_t::boot_loader_name
```

Boot loader name.

3.10.2.5 cmdline

```
u32 multiboot_t::cmdline
```

Command line.

3.10.2.6 config_table

```
u32 multiboot_t::config_table
```

Configuration table.

3.10.2.7 drives_addr

```
u32 multiboot_t::drives_addr
```

Address of drive information.

3.10.2.8 drives_length

```
u32 multiboot_t::drives_length
```

Length of drive information.

3.10.2.9 elf_sec

```
struct multiboot_elf_section_header_table_s multiboot_t::elf_sec
```

ELF section header table.

3.10.2.10 flags

```
u32 multiboot_t::flags
```

Flags indicating available information.

3.10.2.11 mem_lower

```
u32 multiboot_t::mem_lower
```

Lower memory size in KB.

3.10.2.12 mem_upper

```
u32 multiboot_t::mem_upper
```

Upper memory size in KB.

3.10.2.13 mmap_addr

```
u32 multiboot_t::mmap_addr
```

Address of memory map.

3.10.2.14 mmap_length

```
u32 multiboot_t::mmap_length
```

Length of memory map.

3.10.2.15 mods_addr

```
u32 multiboot_t::mods_addr
```

Address of the module list.

3.10.2.16 mods_count

```
u32 multiboot_t::mods_count
```

Number of modules.

3.10.2.17 [union]

```
union { ... } multiboot_t::u
```

Union for symbol table or section header table.

3.10.2.18 vbe_control_info

```
u32 multiboot_t::vbe_control_info
```

VBE control information.

3.10.2.19 vbe_interface_len

```
u16 multiboot_t::vbe_interface_len
```

VBE interface length.

3.10.2.20 vbe_interface_off

```
u16 multiboot_t::vbe_interface_off
```

VBE interface offset.

3.10.2.21 vbe_interface_seg

```
u16 multiboot_t::vbe_interface_seg
```

VBE interface segment.

3.10.2.22 vbe_mode

```
u16 multiboot_t::vbe_mode
```

VBE mode.

3.10.2.23 vbe_mode_info

```
u32 multiboot_t::vbe_mode_info
```

VBE mode information.

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

- multiboot.h

3.11 page_dir_t Struct Reference

```
#include <vmm.h>
```

Data Fields

- u32 entries [TABLES_PER_DIR]

3.11.1 Field Documentation

3.11.1.1 entries

```
u32 page_dir_t::entries[TABLES_PER_DIR]
```

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

- vmm.h

3.12 page_table_t Struct Reference

```
#include <vmm.h>
```

Data Fields

- u32 entries [PAGES_PER_TABLE]

3.12.1 Field Documentation

3.12.1.1 entries

```
u32 page_table_t::entries[PAGES_PER_TABLE]
```

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

- vmm.h

3.13 tss_entry_s Struct Reference

TSS (Task State Segment) entry.

`#include <gdt.h>`

Data Fields

- u32 prev_tss
  previous TSS entry
- u32 esp0
  stack pointer register 0
- u32 ss0
  stack segment register 0
- u32 esp1
  stack pointer register 1
- u32 ss1
  stack segment register 1
- u32 esp2
  stack pointer register 2
- u32 ss2
  stack segment register 2
- u32 cr3
  control register
- u32 eip
  instruction pointer
- u32 eflags
  flags register
- u32 eax
  extended accumulator register
- u32 ecx
  extended counter register
- u32 edx
  extended data register
- u32 ebx
  extended base register
- u32 esp
  extended stack pointer
- u32 ebp
  extended base pointer
- u32 esi
  extended source index
- u32 edi
  extended destination index
- u32 es
  extra segment
- u32 cs
  code segment
- u32 ss
  stack segment

- u32 ds
  data segment
- u32 fs
  additional segment
- u32 gs
  global segment
- u32 ldt
  Local Descriptor Table register.
- u32 trap
  flag in the EFLAGS register
- u32 iomap_base
  input/output map base register

3.13.1 Detailed Description

TSS (Task State Segment) entry.

3.13.2 Field Documentation

3.13.2.1 cr3

```
u32 tss_entry_s::cr3
```

control register

3.13.2.2 cs

```
u32 tss_entry_s::cs
```

code segment

3.13.2.3 ds

```
u32 tss_entry_s::ds
```

data segment

3.13.2.4 eax

```
u32 tss_entry_s::eax
```

extended accumulator register

3.13.2.5 ebp

`u32 tss_entry_s::ebp`

extended base pointer

3.13.2.6 ebx

`u32 tss_entry_s::ebx`

extended base register

3.13.2.7 ecx

`u32 tss_entry_s::ecx`

extended counter register

3.13.2.8 edi

`u32 tss_entry_s::edi`

extended destination index

3.13.2.9 edx

`u32 tss_entry_s::edx`

extended data register

3.13.2.10 eflags

`u32 tss_entry_s::eflags`

flags register

3.13.2.11 eip

`u32 tss_entry_s::eip`

instruction pointer

3.13.2.12 es

`u32 tss_entry_s::es`

extra segment

3.13.2.13 esi

```
u32 tss_entry_s::esi
```

extended source index

3.13.2.14 esp

```
u32 tss_entry_s::esp
```

extended stack pointer

3.13.2.15 esp0

```
u32 tss_entry_s::esp0
```

stack pointer register 0

3.13.2.16 esp1

```
u32 tss_entry_s::esp1
```

stack pointer register 1

3.13.2.17 esp2

```
u32 tss_entry_s::esp2
```

stack pointer register 2

3.13.2.18 fs

```
u32 tss_entry_s::fs
```

additional segment

3.13.2.19 gs

```
u32 tss_entry_s::gs
```

global segment

3.13.2.20 iomap_base

```
u32 tss_entry_s::iomap_base
```

input/output map base register

3.13.2.21 ldt

```
u32 tss_entry_s::ldt
```

Local Descriptor Table register.

3.13.2.22 prev_tss

```
u32 tss_entry_s::prev_tss
```

previous TSS entry

3.13.2.23 ss

```
u32 tss_entry_s::ss
```

stack segment

3.13.2.24 ss0

```
u32 tss_entry_s::ss0
```

stack segment register 0

3.13.2.25 ss1

```
u32 tss_entry_s::ss1
```

stack segment register 1

3.13.2.26 ss2

```
u32 tss_entry_s::ss2
```

stack segment register 2

3.13.2.27 trap

```
u32 tss_entry_s::trap
```

flag in the EFLAGS register

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

- gdt.h

3.14 tty_s Struct Reference

`#include <tty.h>`

Data Fields

- u16 * v_mem
  video memory address
- i32 x_pos
  x position of the cursor
- i32 y_pos
  y position of the cursor
- vga_color_t fg
  foreground color
- vga_color_t bg
  background color
- u8 color
  VGA entry color.
- i32 height
  screen height
- i32 width
  screen width

3.14.1 Field Documentation

3.14.1.1 bg

`vga_color_t tty_s::bg`

background color

3.14.1.2 color

`u8 tty_s::color`

VGA entry color.

3.14.1.3 fg

`vga_color_t tty_s::fg`

foreground color

3.14.1.4 height

`i32 tty_s::height`

screen height

3.14.1.5 v_mem

```
u16* tty_s::v_mem
```

video memory address

3.14.1.6 width

```
i32 tty_s::width
```

screen width

3.14.1.7 x_pos

```
i32 tty_s::x_pos
```

x position of the cursor

3.14.1.8 y_pos

```
i32 tty_s::y_pos
```

y position of the cursor

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

- tty.h

Chapter 4

File Documentation

4.1 ctype.h File Reference

Declares basic character handling functions.

Macros

- #define isalpha(c) (((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z')))
  Checks for an alphabetic character.
- #define isdigit(c) ((c >= '0') && (c <= '9'))
  Checks for a digit.
- #define isalnum(c) (isalpha(c) || isdigit(c))
  Checks for an alphanumeric character.
- #define isascii(c) ((c >= 0) && (c <= 255))
  Checks for an ASCII character.
- #define isprint(c) ((c == ' ') || ((c > 32) && (c < 127)))
  Checks for a printable character (including space).
- #define toupper(c) (islower(c) ? (c - ('a' - 'A')) : c)
  Converts to an uppercase character.
- #define tolower(c) (isupper(c) ? (c + ('a' - 'A')) : c)
  Converts to a lowercase character.
- #define isupper(c) ((c >= 'A') && (c <= 'Z'))
  Checks for an uppercase character.
- #define islower(c) ((c >= 'a') && (c <= 'z'))
  Checks for a lowercase character.

4.1.1 Detailed Description

Declares basic character handling functions.

This file contains declarations for several macros that are useful for testing and mapping characters.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.1.2 Macro Definition Documentation

4.1.2.1 isalnum

```
#define isalnum(
              c ) (isalpha(c) || isdigit(c))
```

Checks for an alphanumeric character.

Parameters

in	*c*	- given character.

Returns

whether c is an alphanumeric character or not.

4.1.2.2 isalpha

```
#define isalpha(
              c ) (((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z')))
```

Checks for an alphabetic character.

Parameters

in	*c*	- given character.

Returns

whether c is an alphabetic character or not.

4.1.2.3 isascii

```
#define isascii(
              c ) ((c >= 0) && (c <= 255))
```

Checks for an ASCII character.

Parameters

in	*c*	- given character.

Returns

whether c is an ASCII character or not.

4.1.2.4 isdigit

```
#define isdigit(
              c ) ((c >= '0') && (c <= '9'))
```

Checks for a digit.

Parameters

in	*c*	- given character.

Returns

whether c is a digit or not.

4.1.2.5 islower

```
#define islower(
              c ) ((c >= 'a') && (c <= 'z'))
```

Checks for a lowercase character.

Parameters

in	*c*	- given character.

Returns

whether c is a lowercase character or not.

4.1.2.6 isprint

```
#define isprint(
              c ) ((c == ' ') || ((c > 32) && (c < 127)))
```

Checks for a printable character (including space).

Parameters

in	*c*	- given character.

Returns

whether c is a printable character (including space) or not.

4.1.2.7 isupper

```
#define isupper(
```

```
            c ) ((c >= 'A') && (c <= 'Z'))
```

Checks for an uppercase character.

Parameters

in	*c*	- given character.

Returns

whether c is an uppercase character or not.

4.1.2.8 tolower

```
#define tolower(
            c ) (isupper(c) ?  (c + ('a' - 'A')) : c)
```

Converts to a lowercase character.

Parameters

in	*c*	- given character.

Returns

lowercase c character.

4.1.2.9 toupper

```
#define toupper(
            c ) (islower(c) ?  (c - ('a' - 'A')) : c)
```

Converts to an uppercase character.

Parameters

in	*c*	- given character.

Returns

uppercase c character.

4.2 ctype.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
```

```
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00035 #ifndef _LIBC_CTYPE_H_
00036 #define _LIBC_CTYPE_H_
00037
00044 #define isalpha(c) (((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z')))
00045
00052 #define isdigit(c) ((c >= '0') && (c <= '9'))
00053
00060 #define isalnum(c) (isalpha(c) || isdigit(c))
00061
00068 #define isascii(c) ((c >= 0) && (c <= 255))
00069
00076 #define isprint(c) ((c == ' ') || ((c > 32) && (c < 127)))
00077
00084 #define toupper(c) (islower(c) ? (c - ('a' - 'A')) : c)
00085
00092 #define tolower(c) (isupper(c) ? (c + ('a' - 'A')) : c)
00093
00100 #define isupper(c) ((c >= 'A') && (c <= 'Z'))
00101
00108 #define islower(c) ((c >= 'a') && (c <= 'z'))
00109
00110
00111 #endif /* _LIBC_CTYPE_H_ */
```

4.3 math.h File Reference

Standard mathematical functions and constants.

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

Macros

- #define PI 3.141592653589793

  The mathematical constant Pi.
- #define E 2.718281828459045

  The mathematical constant e (Euler's number).
- #define _NAN (0.0f / 0.0f)

  Represents a NaN (Not-a-Number) value.
- #define abs(x) ((x) < 0 ? -(x) : (x))

  Calculate the absolute value of a given value.
- #define ceil_div(x, y) (((x + y) - 1) / y)

  Perform ceiling division of two numbers.

Functions

- f64 log (f64 x)

  Calculate natural logarithm.
- f64 pow (f64 x, f64 y)

  Calculate the power of a given base raised to the exponent.
- f64 exp (f64 x)

  Calculate the exponent of given value.
- f64 sqrt (f64 x)

  Calculates the square root of given value.

4.3.1 Detailed Description

Standard mathematical functions and constants.

This file contains declarations for standard mathematical functions and constants. It includes functions for common mathematical operations such as logarithmic functions, exponential functions, and more.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.3.2 Macro Definition Documentation

4.3.2.1 _NAN

```
#define _NAN (0.0f / 0.0f)
```

Represents a NaN (Not-a-Number) value.

This constant represents a NaN value, which is the result of dividing 0.0 by 0.0.

4.3.2.2 abs

```
#define abs(
            x ) ((x) < 0 ? -(x) : (x))
```

Calculate the absolute value of a given value.

This macro calculates the absolute value of the given value.

Parameters

in	*x*	- given value.

Returns

the absolute value of the given value.

4.3.2.3 ceil_div

```
#define ceil_div(
              x,
              y ) (((x + y) - 1) / y)
```

Perform ceiling division of two numbers.

This macro performs ceiling division of two numbers, rounding up to the nearest integer.

Parameters

in	*x*	- the dividend.
in	*y*	- the divisor.

Returns

the result of ceiling division of x by y.

4.3.2.4 E

```
#define E 2.718281828459045
```

The mathematical constant e (Euler's number).

This constant defines the value of the mathematical constant e (approximately 2.718).

4.3.2.5 PI

```
#define PI 3.141592653589793
```

The mathematical constant Pi.

This constant defines the value of Pi (approximately 3.14).

4.3.3 Function Documentation

4.3.3.1 exp()

```
f64 exp (
             f64 x )
```

Calculate the exponent of given value.

Parameters

in	*x*	- given value.

Returns

the exponent of x.

4.3.3.2 log()

```
f64 log (
            f64 x )
```

Calculate natural logarithm.

Parameters

in	*x*	- given value.

Returns

natural logarithm of x.

4.3.3.3 pow()

```
f64 pow (
            f64 x,
            f64 y )
```

Calculate the power of a given base raised to the exponent.

Parameters

in	*x*	- base value.
in	*y*	- exponent value.

Returns

the power of x raised to the y.

4.3.3.4 sqrt()

```
f64 sqrt (
            f64 x )
```

Calculates the square root of given value.

Parameters

in	*x*	- given value.

Returns

the square root of x.

4.4 math.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00036 #ifndef _LIBC_MATH_H_
00037 #define _LIBC_MATH_H_
00038
00039 #include <stdint.h>
00040
00046 #define PI 3.141592653589793
00047
00053 #define E  2.718281828459045
00054
00060 #define _NAN (0.0f / 0.0f)
00061
00070 #define abs(x) ((x) < 0 ? -(x) : (x))
00071
00081 #define ceil_div(x, y) (((x + y) - 1) / y)
00082
00089 f64 log(f64 x);
00090
00098 f64 pow(f64 x, f64 y);
00099
00106 f64 exp(f64 x);
00107
00114 f64 sqrt(f64 x);
00115
00116
00117 #endif /* _LIBC_MATH_H_ */
```

4.5 stdarg.h File Reference

Defines several macros for stepping through a list of arguments.

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

Macros

- #define va_start(v, l) __builtin_va_start(v,l)

  This macro initializes ap for subsequent use by va_arg() and va_end(), and must be called first.
- #define va_arg(v, l) __builtin_va_arg(v,l)

  This macro expands to an expression that has the type and value of the next argument in the call.
- #define va_copy(d, s) __builtin_va_copy(d,s)

  This macro copies the (previously initialized) variable argument list src to dest.
- #define va_end(v) __builtin_va_end(v)

  Each invocation of va_start() must be matched by a corresponding invocation of va_end() in the same function.

Typedefs

- typedef __builtin_va_list va_list

4.5.1 Detailed Description

Defines several macros for stepping through a list of arguments.

This file contains declarations for several macros that are useful for managing functions with variable number of arguments.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.5.2 Macro Definition Documentation

4.5.2.1 va_arg

```
#define va_arg(
              v,
              l ) __builtin_va_arg(v,l)
```

This macro expands to an expression that has the type and value of the next argument in the call.

The argument ap is the va_list ap initialized by va_start(). Each call to va_arg() modifies ap so that the next call returns the next argument. The argument type is a type name specified so that the type of a pointer to an object that has the specified type can be obtained simply by adding a $*$ to type.

The first use of the va_arg() macro after that of the va_start() macro returns the argument after last. Successive invocations return the values of the remaining arguments.

If there is no next argument, or if type is not compatible with the type of the actual next argument (as promoted accord- ing to the default argument promotions), random errors will occur.

If ap is passed to a function that uses va_arg(ap,type), then the value of ap is undefined after the return of that function.

Parameters

in	*v*	- arguments list.
in	*l*	- number of arguments.

4.5.2.2 va_copy

```
#define va_copy(
              d,
              s ) __builtin_va_copy(d,s)
```

This macro copies the (previously initialized) variable argument list src to dest.

The behavior is as if va_start() were applied to dest with the same last argument, followed by the same number of va_arg() invocations that was used to reach the current state of src.

Parameters

in	*v*	- arguments list.
in	*l*	- number of arguments.

4.5.2.3 va_end

```
#define va_end(
              v ) __builtin_va_end(v)
```

Each invocation of va_start() must be matched by a corresponding invocation of va_end() in the same function.

After the call va_end(ap) the variable ap is undefined. Multiple traversals of the list, each bracketed by va_start() and va_end() are possible. va_end() may be a macro or a function.

Parameters

in	*v*	- arguments list.
in	*l*	- number of arguments.

4.5.2.4 va_start

```
#define va_start(
              v,
              l ) __builtin_va_start(v,l)
```

This macro initializes ap for subsequent use by va_arg() and va_end(), and must be called first.

The argument last is the name of the last argument before the variable argument list, that is, the last argument of which the calling function knows the type.

Because the address of this argument may be used in the va_start() macro, it should not be declared as a register variable, or as a function or an array type.

Parameters

in	*v*	- arguments list.
in	*l*	- number of arguments.

4.5.3 Typedef Documentation

4.5.3.1 va_list

```
typedef __builtin_va_list va_list
```

4.6 stdarg.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00035 #ifndef _LIBC_STDARG_H_
00036 #define _LIBC_STDARG_H_
00037
00038 #include <stddef.h>
00039
00055 #define va_start(v,l) __builtin_va_start(v,l)
00056
00082 #define va_arg(v,l)   __builtin_va_arg(v,l)
00083
00095 #define va_copy(d,s)  __builtin_va_copy(d,s)
00096
00108 #define va_end(v)     __builtin_va_end(v)
00109
00110 typedef __builtin_va_list va_list;
00111
00112
00113 #endif /* _LIBC_STDARG_H_ */
```

4.7 stddef.h File Reference

Standard defenitions and constants.

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

Macros

- #define NULL ((void *)0)
- #define usize u64

4.7.1 Detailed Description

Standard defenitions and constants.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.7.2 Macro Definition Documentation

4.7.2.1 NULL

```
#define NULL ((void *)0)
```

4.7.2.2 usize

```
#define usize u64
```

4.8 stddef.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00031 #ifndef _LIBC_STDDEF_H_
00032 #define _LIBC_STDDEF_H_
00033
00034 #include <stdint.h>
00035
00036 #define NULL   ((void *)0)
00037 #define usize  u64
00038
00039 #endif /* _LIBC_STDDEF_H_ */
```

4.9 stdint.h File Reference

Defines an integer types of a fixed width.

Typedefs

- typedef unsigned long u64
- typedef unsigned int u32
- typedef unsigned short u16
- typedef unsigned char u8
- typedef long i64
- typedef int i32
- typedef short i16
- typedef char i8
- typedef double f64
- typedef float f32

4.9.1 Detailed Description

Defines an integer types of a fixed width.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.9.2 Typedef Documentation

4.9.2.1 f32

```
typedef float f32
```

4.9.2.2 f64

```
typedef double f64
```

4.9.2.3 i16

```
typedef short i16
```

4.9.2.4 i32

```
typedef int i32
```

4.9.2.5 i64

```
typedef long i64
```

4.9.2.6 i8

```
typedef char i8
```

4.9.2.7 u16

```
typedef unsigned short u16
```

4.9.2.8 u32

```
typedef unsigned int u32
```

4.9.2.9 u64

```
typedef unsigned long u64
```

4.9.2.10 u8

```
typedef unsigned char u8
```

4.10 stdint.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00031 #ifndef _LIBC_STDINT_H_
00032 #define _LIBC_STDINT_H_
00033
00034 /* unsigned values */
00035 typedef unsigned long   u64;
00036 typedef unsigned int    u32;
00037 typedef unsigned short  u16;
00038 typedef unsigned char   u8;
00039
00040 /* signed values */
00041 typedef long            i64;
00042 typedef int             i32;
00043 typedef short           i16;
00044 typedef char            i8;
00045
00046 typedef double          f64;
00047 typedef float           f32;
00048
00049 #endif /* _LIBC_STDINT_H_ */
```

4.11 stdio.h File Reference

Standard input/output functions.

```
#include <nos/vga.h>
#include <stdarg.h>
```

Functions

- void putk (const char *str)

  Print given string on the same line.
- void cputk (const char *str, vga_color_t fg, vga_color_t bg)

  Print colored string and a trailing newline.
- void puts (const char *str)

  Print string and a trailing newline.
- void vsnprintf (char *buf, usize size, const char *fmt, va_list args)

  Formats and prints data to buffer.

4.11.1 Detailed Description

Standard input/output functions.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.11.2 Function Documentation

4.11.2.1 cputk()

```
void cputk (
            const char * str,
            vga_color_t fg,
            vga_color_t bg )
```

Print colored string and a trailing newline.

Parameters

in	*str*	- given null terminated string
in	*fg*	- given foreground color.
in	*bg*	- given background color.

4.11.2.2 putk()

```
void putk (
            const char * str )
```

Print given string on the same line.

Parameters

in	*str*	- given null terminated string

4.11.2.3 puts()

```
void puts (
            const char * str )
```

Print string and a trailing newline.

Parameters

in	*str*	- given null terminated string

4.11.2.4 vsnprintf()

```
void vsnprintf (
            char * buf,
            usize size,
            const char * fmt,
            va_list args )
```

Formats and prints data to buffer.

Parameters

out	*buf*	- given buffer for containing formated result.
in	*size*	- given buffer size.
in	*fmt*	- given format string.
in	*args*	- given variable list of arguments.

4.12 stdio.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
```

```
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00031 #ifndef _LIBC_STDIO_H_
00032 #define _LIBC_STDIO_H_
00033
00034 #include <nos/vga.h>
00035 #include <stdarg.h>
00036
00042 void putk(const char *str);
00043
00051 void cputk(const char *str, vga_color_t fg, vga_color_t bg);
00052
00058 void puts(const char *str);
00059
00068 void vsnprintf(char *buf, usize size, const char *fmt, va_list args);
00069
00070
00071 #endif /* _LIBC_STDIO_H_ */
```

4.13 string.h File Reference

Defines several strings and memory management functions.

`#include <stddef.h>`

Functions

- i32 strlen (const char *str)

  Get string length.
- i32 strncmp (const char *s1, const char *s2, usize n)

  Compares the two strings s1 and s2.
- usize strncpy (char *dest, const char *src, usize size)

  Copy a string with truncation.
- usize strncat (char *dest, const char *src, usize size)

  Catenate a string with truncation.
- void * memset (void *s, i32 c, usize n)

  Fills the first n bytes of the memory of the area pointed to by s with the constant byte c.
- void bzero (void *s, usize n)

  Erases the data in the n bytes of the memory of the area pointed to by s, by writing '\0' bytes to that area.
- void * memcpy (void *dest, const void *src, usize n)

  Copies n bytes from memory area src to memory area dest.
- i32 memcmp (const void *s1, const void *s2, usize n)

  Compares the first n bytes (each interpreted as unsigned char) of the memory areas s1 and s2.

4.13.1 Detailed Description

Defines several strings and memory management functions.

This header file provides functions for manipulating strings, such as copying, concatenating and comparing.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.13.2 Function Documentation

4.13.2.1 bzero()

```
void bzero (
            void * s,
            usize n )
```

Erases the data in the n bytes of the memory of the area pointed to by s, by writing '\0' bytes to that area.

Parameters

out	*s*	- given buffer pointer.
in	*n*	- given number of buffer bytes to erase.

4.13.2.2 memcmp()

```
i32 memcmp (
            const void * s1,
            const void * s2,
            usize n )
```

Compares the first n bytes (each interpreted as unsigned char) of the memory areas s1 and s2.

Parameters

in	*s1*	- first given memory area pointer.
in	*s2*	- second given memory area pointer.
in	*n*	- given number of bytes to compare.

Returns

0, if s1 and s2 are equal;

a negative value if s1 is less than s2;

a positive value if s1 is greater than s2.

4.13.2.3 memcpy()

```
void * memcpy (
            void * dest,
            const void * src,
            usize n )
```

Copies n bytes from memory area src to memory area dest.

Parameters

out	*dest*	- given destination buffer.
in	*src*	- given source buffer.
in	*n*	- given number of bytes to copy.

Returns

destination buffer pointer.

4.13.2.4 memset()

```
void * memset (
            void * s,
            i32 c,
            usize n )
```

Fills the first n bytes of the memory of the area pointed to by s with the constant byte c.

Parameters

out	*s*	- given buffer pointer.
in	*c*	- given byte for filling buffer.
in	*n*	- given number of buffer bytes to fill.

Returns

filled buffer pointer.

4.13.2.5 strlen()

```
i32 strlen (
            const char * str )
```

Get string length.

Parameters

in	*str*	- given null terminated string.

Returns

str length.

4.13.2.6 strncat()

```
usize strncat (
            char * dest,
            const char * src,
            usize size )
```

Catenate a string with truncation.

Parameters

out	*dest*	- given buffer for concatenated string.
in	*src*	- given source null terminated string.
in	*size*	- given size to concatenate.

Returns

length of new concatenated string.

4.13.2.7 strncmp()

```
i32 strncmp (
            const char * s1,
            const char * s2,
            usize n )
```

Compares the two strings s1 and s2.

Parameters

in	*s1*	- first given null terminated string.
in	*s2*	- second given null terminated string.
in	*n*	- given number of symbols for comparison.

Returns

0, if s1 and s2 are equal;

a negative value if s1 is less than s2;

a positive value if s1 is greater than s2.

4.13.2.8 strncpy()

```
usize strncpy (
            char * dest,
            const char * src,
            usize size )
```

Copy a string with truncation.

Parameters

out	*dest*	- given buffer for copied string.
in	*src*	- given source null terminated string.
in	*size*	- given size to copy.

Returns

number of copied string characters.

4.14 string.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _LIBC_STRING_H_
00035 #define _LIBC_STRING_H_
00036
00037 #include <stddef.h>
00038
00045 i32   strlen(const char *str);
00046
00057 i32   strncmp(const char *s1, const char *s2, usize n);
00058
00067 usize strncpy(char *dest, const char *src, usize size);
00068
00077 usize strncat(char *dest, const char *src, usize size);
00078
00088 void  *memset(void *s, i32 c, usize n);
00089
00098 void  bzero(void *s, usize n);
00099
00108 void  *memcpy(void *dest, const void *src, usize n);
00109
00121 i32   memcmp(const void *s1, const void *s2, usize n);
00122
00123 #endif /* _LIBC_STRING_H_ */
```

4.15 gdt.h File Reference

Contains GDT and TSS structures and management functions.

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

Data Structures

- struct gdt_entry_s
  GDT entry (segment descriptor) - tells the CPU the attributes of a given segment.
- struct gdt_ptr_s
  GDT pointer structure.
- struct tss_entry_s
  TSS (Task State Segment) entry.

Typedefs

- typedef struct gdt_entry_s gdt_entry_t
- typedef struct gdt_ptr_s gdt_ptr_t
- typedef struct tss_entry_s tss_entry_t

Functions

- struct gdt_entry_s __attribute__ ((packed))
  prevent the compiler from optimizing
- void gdt_init (void)
  initialize Global Descriptor Table
- void set_gdt_gate (u32 num, u32 base, u32 limit, u8 access, u8 gran)
  Set the Global Descriptor Table content.
- void tss_write (u32 num, u16 ss0, u32 esp0)
  Write Task State Segment (TSS) entry.

Variables

- u16 limit
- u16 base_low
- u8 base_mid
- u8 access
- u8 flags
- u8 base_high
- u32 base
- u32 prev_tss
  previous TSS entry
- u32 esp0
  stack pointer register 0
- u32 ss0
  stack segment register 0
- u32 esp1

stack pointer register 1

- u32 ss1

  stack segment register 1
- u32 esp2

  stack pointer register 2
- u32 ss2

  stack segment register 2
- u32 cr3

  control register
- u32 eip

  instruction pointer
- u32 eflags

  flags register
- u32 eax

  extended accumulator register
- u32 ecx

  extended counter register
- u32 edx

  extended data register
- u32 ebx

  extended base register
- u32 esp

  extended stack pointer
- u32 ebp

  extended base pointer
- u32 esi

  extended source index
- u32 edi

  extended destination index
- u32 es

  extra segment
- u32 cs

  code segment
- u32 ss

  stack segment
- u32 ds

  data segment
- u32 fs

  additional segment
- u32 gs

  global segment
- u32 ldt

  Local Descriptor Table register.
- u32 trap

  flag in the EFLAGS register
- u32 iomap_base

  input/output map base register

4.15.1 Detailed Description

Contains GDT and TSS structures and management functions.

Global Descriptor Table (GDT) is a structure specific to the IA-32 and x86-64 architectures. It contains entries telling the CPU about memory segments.

The TSS is used for hardware task switching and contains information about the state of a task.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.15.2 Typedef Documentation

4.15.2.1 gdt_entry_t

```
typedef struct gdt_entry_s gdt_entry_t
```

4.15.2.2 gdt_ptr_t

```
typedef struct gdt_ptr_s gdt_ptr_t
```

4.15.2.3 tss_entry_t

```
typedef struct tss_entry_s tss_entry_t
```

4.15.3 Function Documentation

4.15.3.1 __attribute__()

```
struct idt_ptr_s __attribute__ (
             (packed)  )
```

prevent the compiler from optimizing

4.15.3.2 gdt_init()

```
void gdt_init (
             void  )
```

initialize Global Descriptor Table

4.15.3.3 set_gdt_gate()

```
void set_gdt_gate (
             u32 num,
             u32 base,
             u32 limit,
             u8 access,
             u8 gran )
```

Set the Global Descriptor Table content.

Parameters

in	*num*	- given GDT entry number.
in	*base*	- given base address of the segment.
in	*limit*	- given limit of the segment.
in	*access*	- given access byte for the segment.
in	*gran*	- given granularity and size information.

4.15.3.4 tss_write()

```
void tss_write (
            u32 num,
            u16 ss0,
            u32 esp0 )
```

Write Task State Segment (TSS) entry.

Parameters

in	*num*	- given TSS entry number.
in	*ss0*	- given stack segment for privilege level 0.
in	*esp0*	- given stack pointer for privilege level 0.

4.15.4 Variable Documentation

4.15.4.1 access

```
u8 access
```

4.15.4.2 base

```
u32 base
```

4.15.4.3 base_high

```
u8 base_high
```

4.15.4.4 base_low

```
u16 base_low
```

4.15.4.5 base_mid

```
u8 base_mid
```

4.15.4.6 cr3

```
u32 cr3
```

control register

4.15.4.7 cs

```
u32 cs
```

code segment

4.15.4.8 ds

```
u32 ds
```

data segment

4.15.4.9 eax

```
u32 eax
```

extended accumulator register

4.15.4.10 ebp

```
u32 ebp
```

extended base pointer

4.15.4.11 ebx

```
u32 ebx
```

extended base register

4.15.4.12 ecx

```
u32 ecx
```

extended counter register

4.15.4.13 edi

```
u32 edi
```

extended destination index

4.15.4.14 edx

```
u32 edx
```

extended data register

4.15.4.15 eflags

```
u32 eflags
```

flags register

4.15.4.16 eip

```
u32 eip
```

instruction pointer

4.15.4.17 es

```
u32 es
```

extra segment

4.15.4.18 esi

```
u32 esi
```

extended source index

4.15.4.19 esp

```
u32 esp
```

extended stack pointer

4.15.4.20 esp0

```
u32 esp0
```

stack pointer register 0

4.15.4.21 esp1

```
u32 esp1
```

stack pointer register 1

4.15.4.22 esp2

```
u32 esp2
```

stack pointer register 2

4.15.4.23 flags

```
u8 flags
```

4.15.4.24 fs

```
u32 fs
```

additional segment

4.15.4.25 gs

```
u32 gs
```

global segment

4.15.4.26 iomap_base

```
u32 iomap_base
```

input/output map base register

4.15.4.27 ldt

```
u32 ldt
```

Local Descriptor Table register.

4.15.4.28 limit

```
u16 limit
```

4.15.4.29 prev_tss

```
u32 prev_tss
```

previous TSS entry

4.15.4.30 ss

```
u32 ss
```

stack segment

4.15.4.31 ss0

```
u32 ss0
```

stack segment register 0

4.15.4.32 ss1

```
u32 ss1
```

stack segment register 1

4.15.4.33 ss2

```
u32 ss2
```

stack segment register 2

4.15.4.34 trap

```
u32 trap
```

flag in the EFLAGS register

4.16 gdt.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00038 #ifndef _NOS_KERNEL_GDT_H_
```

```
00039 #define _NOS_KERNEL_GDT_H_
00040
00041 #include <stdint.h>
00042
00048 struct gdt_entry_s {
00049     u16 limit;
00050     u16 base_low;
00051     u8  base_mid;
00052     u8  access;
00053     u8  flags;
00054     u8  base_high;
00055 } __attribute__((packed)); /* prevent the compiler from optimizing */
00056
00057 typedef struct gdt_entry_s gdt_entry_t;
00058
00060 struct gdt_ptr_s {
00061     u16 limit;
00062     u32 base;
00063 } __attribute__((packed)); /* prevent the compiler from optimizing */
00064
00065 typedef struct gdt_ptr_s gdt_ptr_t;
00066
00068 struct tss_entry_s {
00069     u32 prev_tss;
00070     u32 esp0;
00071     u32 ss0;
00072     u32 esp1;
00073     u32 ss1;
00074     u32 esp2;
00075     u32 ss2;
00076     u32 cr3;
00077     u32 eip;
00078     u32 eflags;
00079     u32 eax;
00080     u32 ecx;
00081     u32 edx;
00082     u32 ebx;
00083     u32 esp;
00084     u32 ebp;
00085     u32 esi;
00086     u32 edi;
00087     u32 es;
00088     u32 cs;
00089     u32 ss;
00090     u32 ds;
00091     u32 fs;
00092     u32 gs;
00093     u32 ldt;
00094     u32 trap;
00095     u32 iomap_base;
00096 } __attribute__((packed));
00097
00098 typedef struct tss_entry_s tss_entry_t;
00099
00101 void gdt_init(void);
00102
00112 void set_gdt_gate(u32 num, u32 base, u32 limit, u8 access, u8 gran);
00113
00121 void tss_write(u32 num, u16 ss0, u32 esp0);
00122
00123 #endif /* _NOS_KERNEL_GDT_H_ */
```

4.17 idt.h File Reference

Contains IDT structures and management functions.

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

Data Structures

- struct idt_entry_s

  Interrupt Descriptor Table entry structure.
- struct idt_ptr_s

  Interrupt Descriptor Table pointer structure.

Typedefs

- typedef struct idt_entry_s idt_entry_t
- typedef struct idt_ptr_s idt_ptr_t

Functions

- struct idt_entry_s __attribute__ ((packed))
- void idt_init (void)

  initialize Interrupt Descriptor Table.
- void set_idt_gate (u8 num, u32 base, u16 sel, u8 flags)

  Set Interrupt Descriptor Table (IDT) gate.

Variables

- u16 base_low
- u16 sel
- u8 always0
- u8 flags
- u16 base_high
- u16 limit
- u32 base

4.17.1 Detailed Description

Contains IDT structures and management functions.

IDT (Interrupt Descriptor Table) - telling the CPU where the Interrupt Service Routines (ISR) are located.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.17.2 Typedef Documentation

4.17.2.1 idt_entry_t

```
typedef struct idt_entry_s idt_entry_t
```

4.17.2.2 idt_ptr_t

```
typedef struct idt_ptr_s idt_ptr_t
```

4.17.3 Function Documentation

4.17.3.1 __attribute__()

```
struct idt_entry_s __attribute__ (
             (packed)  )
```

4.17.3.2 idt_init()

```
void idt_init (
             void  )
```

initialize Interrupt Descriptor Table.

4.17.3.3 set_idt_gate()

```
void set_idt_gate (
             u8 num,
             u32 base,
             u16 sel,
             u8 flags )
```

Set Interrupt Descriptor Table (IDT) gate.

Parameters

in	*num*	- given IDT entry number.
in	*base*	- given base address of the interrupt handler.
in	*sel*	- given code segment selector.
in	*flags*	- given flags and type of the interrupt gate.

4.17.4 Variable Documentation

4.17.4.1 always0

```
u8 always0
```

4.17.4.2 base

```
u32 base
```

4.17.4.3 base_high

```
u16 base_high
```

4.17.4.4 base_low

```
u16 base_low
```

4.17.4.5 flags

```
u8 flags
```

4.17.4.6 limit

```
u16 limit
```

4.17.4.7 sel

```
u16 sel
```

4.18 idt.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00035 #ifndef _NOS_KERNEL_IDT_H_
00036 #define _NOS_KERNEL_IDT_H_
00037
00038 #include <stdint.h>
00039
00041 struct idt_entry_s {
00042     u16 base_low;
00043     u16 sel;
00044     u8  always0;
00045     u8  flags;
00046     u16 base_high;
00047 } __attribute__((packed)); /* prevent the compiler from optimizing */
00048
00049 typedef struct idt_entry_s idt_entry_t;
00050
00052 struct idt_ptr_s {
00053     u16 limit;
00054     u32 base;
00055 } __attribute__((packed)); /* prevent the compiler from optimizing */
00056
00057 typedef struct idt_ptr_s idt_ptr_t;
00058
00060 void idt_init(void);
00061
00070 void set_idt_gate(u8 num, u32 base, u16 sel, u8 flags);
00071
00072 #endif /* _NOS_KERNEL_IDT_H_ */
```

4.19 irq.h File Reference

IRQ (Interrupt Request). Contains definitions related to interrupt handling.

```
#include <stdint.h>
#include <nos/irq.h>
```

Data Structures

- struct int_reg_s

  Structure representing interrupt register state.

Typedefs

- typedef struct int_reg_s int_reg_t

  Typedef for int_reg_s.
- typedef void(∗ irq_handler_t) (int_reg_t ∗)

  Typedef for IRQ handler function pointer.

Functions

- struct int_reg_s __attribute__ ((packed))
- void isr_handler (int_reg_t ∗regs)

  ISR handler function.
- void irq_handler (int_reg_t ∗regs)

  IRQ handler function.
- void irq_install_handler (int irq, irq_handler_t handler)

  Install handler for IRQ.
- void irq_uninstall_handler (int irq)

  Uninstall handler for IRQ.
- void isr0 (void)

  Declare ISR functions for hardware interrupts 0-31.
- void isr1 (void)
- void isr2 (void)
- void isr3 (void)
- void isr4 (void)
- void isr5 (void)
- void isr6 (void)
- void isr7 (void)
- void isr8 (void)
- void isr9 (void)
- void isr10 (void)
- void isr11 (void)
- void isr12 (void)
- void isr13 (void)
- void isr14 (void)
- void isr15 (void)
- void isr16 (void)
- void isr17 (void)

- void isr18 (void)
- void isr19 (void)
- void isr20 (void)
- void isr21 (void)
- void isr22 (void)
- void isr23 (void)
- void isr24 (void)
- void isr25 (void)
- void isr26 (void)
- void isr27 (void)
- void isr28 (void)
- void isr29 (void)
- void isr30 (void)
- void isr31 (void)
- void isr128 (void)

  Declare ISR functions for system calls.
- void isr177 (void)
- void irq0 (void)

  Declare ISR functions for hardware interrupts 0-15.
- void irq1 (void)
- void irq2 (void)
- void irq3 (void)
- void irq4 (void)
- void irq5 (void)
- void irq6 (void)
- void irq7 (void)
- void irq8 (void)
- void irq9 (void)
- void irq10 (void)
- void irq11 (void)
- void irq12 (void)
- void irq13 (void)
- void irq14 (void)
- void irq15 (void)

Variables

- u32 cr2

  Control Register 2.
- u32 ds

  Data Segment.
- u32 edi

  Destination Index.
- u32 esi

  Source Index.
- u32 ebp

  Base Pointer.
- u32 esp

  Stack Pointer.
- u32 ebx

  Base Register.
- u32 edx

Data Register.

- u32 ecx

  Counter Register.
- u32 eax

  Accumulator Register.
- u32 int_no

  Interrupt Number.
- u32 err_code

  Error Code.
- u32 eip

  Instruction Pointer.
- u32 csm

  Code Segment.
- u32 eflags

  Flags Register.
- u32 useresp

  User Stack Pointer.
- u32 ss

  Stack Segment.

4.19.1 Detailed Description

IRQ (Interrupt Request). Contains definitions related to interrupt handling.

This header file includes constants, structures, and functions related to managing interrupts in the kernel.

ISR (Interrupt Service Routine). An ISR is a software function or routine that is executed in response to an interrupt generated by hardware or software.

ISRs are used to manage various types of interrupts, such as hardware interrupts from devices like keyboards or timers, or software interrupts triggered by specific instructions.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.19.2 Typedef Documentation

4.19.2.1 int_reg_t

```
typedef struct int_reg_s int_reg_t
```

Typedef for int_reg_s.

4.19.2.2 irq_handler_t

```
typedef void(* irq_handler_t) (int_reg_t *)
```

Typedef for IRQ handler function pointer.

4.19.3 Function Documentation

4.19.3.1 __attribute__()

```
struct int_reg_s __attribute__ (
             (packed)  )
```

4.19.3.2 irq0()

```
void irq0 (
             void  )  [extern]
```

Declare ISR functions for hardware interrupts 0-15.

4.19.3.3 irq1()

```
void irq1 (
             void  )  [extern]
```

4.19.3.4 irq10()

```
void irq10 (
             void  )  [extern]
```

4.19.3.5 irq11()

```
void irq11 (
             void  )  [extern]
```

4.19.3.6 irq12()

```
void irq12 (
             void  )  [extern]
```

4.19.3.7 irq13()

```
void irq13 (
             void  )  [extern]
```

4.19.3.8 irq14()

```
void irq14 (
            void  )  [extern]
```

4.19.3.9 irq15()

```
void irq15 (
            void  )  [extern]
```

4.19.3.10 irq2()

```
void irq2 (
            void  )  [extern]
```

4.19.3.11 irq3()

```
void irq3 (
            void  )  [extern]
```

4.19.3.12 irq4()

```
void irq4 (
            void  )  [extern]
```

4.19.3.13 irq5()

```
void irq5 (
            void  )  [extern]
```

4.19.3.14 irq6()

```
void irq6 (
            void  )  [extern]
```

4.19.3.15 irq7()

```
void irq7 (
            void  )  [extern]
```

4.19.3.16 irq8()

```
void irq8 (
            void  )  [extern]
```

4.19.3.17 irq9()

```
void irq9 (
            void  )  [extern]
```

4.19.3.18 irq_handler()

```
void irq_handler (
            int_reg_t * regs )
```

IRQ handler function.

Parameters

in	*regs*	- given pointer to interrupt register state.

4.19.3.19 irq_install_handler()

```
void irq_install_handler (
            int irq,
            irq_handler_t handler )
```

Install handler for IRQ.

Parameters

in	*irq*	- given IRQ number.
in	*handler*	- given pointer to IRQ handler function.

4.19.3.20 irq_uninstall_handler()

```
void irq_uninstall_handler (
            int irq )
```

Uninstall handler for IRQ.

Parameters

in	*irq*	- given IRQ number.

4.19.3.21 isr0()

```
void isr0 (
            void  )  [extern]
```

Declare ISR functions for hardware interrupts 0-31.

4.19.3.22 isr1()

```
void isr1 (
            void  )  [extern]
```

4.19.3.23 isr10()

```
void isr10 (
            void  )  [extern]
```

4.19.3.24 isr11()

```
void isr11 (
            void  )  [extern]
```

4.19.3.25 isr12()

```
void isr12 (
            void  )  [extern]
```

4.19.3.26 isr128()

```
void isr128 (
            void  )  [extern]
```

Declare ISR functions for system calls.

4.19.3.27 isr13()

```
void isr13 (
            void  )  [extern]
```

4.19.3.28 isr14()

```
void isr14 (
            void  )  [extern]
```

4.19.3.29 isr15()

```
void isr15 (
            void  )  [extern]
```

4.19.3.30 isr16()

```
void isr16 (
            void  )  [extern]
```

4.19.3.31 isr17()

```
void isr17 (
            void  )  [extern]
```

4.19.3.32 isr177()

```
void isr177 (
            void  )  [extern]
```

4.19.3.33 isr18()

```
void isr18 (
            void  )  [extern]
```

4.19.3.34 isr19()

```
void isr19 (
            void  )  [extern]
```

4.19.3.35 isr2()

```
void isr2 (
            void  )  [extern]
```

4.19.3.36 isr20()

```
void isr20 (
            void  )  [extern]
```

4.19.3.37 isr21()

```
void isr21 (
            void  )  [extern]
```

4.19.3.38 isr22()

```
void isr22 (
            void  )  [extern]
```

4.19.3.39 isr23()

```
void isr23 (
            void  )  [extern]
```

4.19.3.40 isr24()

```
void isr24 (
            void  )  [extern]
```

4.19.3.41 isr25()

```
void isr25 (
            void  )  [extern]
```

4.19.3.42 isr26()

```
void isr26 (
            void  )  [extern]
```

4.19.3.43 isr27()

```
void isr27 (
            void  )  [extern]
```

4.19.3.44 isr28()

```
void isr28 (
            void  )  [extern]
```

4.19.3.45 isr29()

```
void isr29 (
            void  )  [extern]
```

4.19.3.46 isr3()

```
void isr3 (
            void  )  [extern]
```

4.19.3.47 isr30()

```
void isr30 (
            void  )  [extern]
```

4.19.3.48 isr31()

```
void isr31 (
            void  )  [extern]
```

4.19.3.49 isr4()

```
void isr4 (
            void  )  [extern]
```

4.19.3.50 isr5()

```
void isr5 (
            void  )  [extern]
```

4.19.3.51 isr6()

```
void isr6 (
            void  )  [extern]
```

4.19.3.52 isr7()

```
void isr7 (
            void  )  [extern]
```

4.19.3.53 isr8()

```
void isr8 (
            void  )  [extern]
```

4.19.3.54 isr9()

```
void isr9 (
            void  )  [extern]
```

4.19.3.55 isr_handler()

```
void isr_handler (
            int_reg_t * regs )
```

ISR handler function.

Parameters

in	*regs*	- given pointer to interrupt register state.

4.19.4 Variable Documentation

4.19.4.1 cr2

```
u32 cr2
```

Control Register 2.

4.19.4.2 csm

```
u32 csm
```

Code Segment.

4.19.4.3 ds

```
u32 ds
```

Data Segment.

4.19.4.4 eax

```
u32 eax
```

Accumulator Register.

4.19.4.5 ebp

```
u32 ebp
```

Base Pointer.

4.19.4.6 ebx

```
u32 ebx
```

Base Register.

4.19.4.7 ecx

```
u32 ecx
```

Counter Register.

4.19.4.8 edi

```
u32 edi
```

Destination Index.

4.19.4.9 edx

```
u32 edx
```

Data Register.

4.19.4.10 eflags

```
u32 eflags
```

Flags Register.

4.19.4.11 eip

```
u32 eip
```

Instruction Pointer.

4.19.4.12 err_code

```
u32 err_code
```

Error Code.

4.19.4.13 esi

```
u32 esi
```

Source Index.

4.19.4.14 esp

```
u32 esp
```

Stack Pointer.

4.19.4.15 int_no

```
u32 int_no
```

Interrupt Number.

4.19.4.16 ss

```
u32 ss
```

Stack Segment.

4.19.4.17 useresp

```
u32 useresp
```

User Stack Pointer.

4.20 irq.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00041 #ifndef _NOS_KERNEL_IRQ_H_
00042 #define _NOS_KERNEL_IRQ_H_
00043
00044 #include <stdint.h>
00045
00046 #include <nos/irq.h>
00047
00049 struct int_reg_s {
00050     u32 cr2;
00051     u32 ds;
00052     u32 edi;
00053     u32 esi;
00054     u32 ebp;
00055     u32 esp;
00056     u32 ebx;
00057     u32 edx;
00058     u32 ecx;
00059     u32 eax;
00060     u32 int_no;
00061     u32 err_code;
00062     u32 eip;
00063     u32 csm;
00064     u32 eflags;
00065     u32 useresp;
```

```
00066     u32 ss;
00067 } __attribute__((packed)); /*Prevent the compiler from optimizing*/
00068
00070 typedef struct int_reg_s int_reg_t;
00071
00073 typedef void (*irq_handler_t) (int_reg_t *);
00074
00080 void isr_handler(int_reg_t *regs);
00081
00087 void irq_handler(int_reg_t *regs);
00088
00095 void irq_install_handler(int irq, irq_handler_t handler);
00096
00102 void irq_uninstall_handler(int irq);
00103
00105 extern void isr0(void);
00106 extern void isr1(void);
00107 extern void isr2(void);
00108 extern void isr3(void);
00109 extern void isr4(void);
00110 extern void isr5(void);
00111 extern void isr6(void);
00112 extern void isr7(void);
00113 extern void isr8(void);
00114 extern void isr9(void);
00115 extern void isr10(void);
00116 extern void isr11(void);
00117 extern void isr12(void);
00118 extern void isr13(void);
00119 extern void isr14(void);
00120 extern void isr15(void);
00121 extern void isr16(void);
00122 extern void isr17(void);
00123 extern void isr18(void);
00124 extern void isr19(void);
00125 extern void isr20(void);
00126 extern void isr21(void);
00127 extern void isr22(void);
00128 extern void isr23(void);
00129 extern void isr24(void);
00130 extern void isr25(void);
00131 extern void isr26(void);
00132 extern void isr27(void);
00133 extern void isr28(void);
00134 extern void isr29(void);
00135 extern void isr30(void);
00136 extern void isr31(void);
00137
00139 extern void isr128(void);
00140 extern void isr177(void);
00141
00143 extern void irq0(void);
00144 extern void irq1(void);
00145 extern void irq2(void);
00146 extern void irq3(void);
00147 extern void irq4(void);
00148 extern void irq5(void);
00149 extern void irq6(void);
00150 extern void irq7(void);
00151 extern void irq8(void);
00152 extern void irq9(void);
00153 extern void irq10(void);
00154 extern void irq11(void);
00155 extern void irq12(void);
00156 extern void irq13(void);
00157 extern void irq14(void);
00158 extern void irq15(void);
00159
00160 #endif /* _NOS_KERNEL_IRQ_H_ */
```

4.21 kernel.h File Reference

Contains declarations for kernel functions and structures.

```
#include <stdint.h>
#include <nos/multiboot.h>
#include <nos/tty.h>
```

Macros

- #define __OS_NAME__ "nos"

  OS information definitions.
- #define __OS_VERSION__ "v0.0.2"
- #define __OS_ARCH__ "x86_32"
- #define __OS_BUILD_DATE__ __DATE__
- #define __OS_BUILD_TIME__ __TIME__
- #define __OS_INFO_FMT__ " %s (%s %s) (c) @alkuzin - 2024\n"
- #define __OS_BUILD_INFO_FMT__ "%s %s <%s>\n"
- #define __DISPLAY_OS_INFO() printk(__OS_INFO_FMT__, __OS_NAME__, __OS_VERSION__, __OS_ARCH__)

  Macro for displaying main OS info: name, version and architecture.
- #define __DISPLAY_OS_BUILD_INFO() printk(__OS_BUILD_INFO_FMT__, " Build time:", __OS_BUILD_TIME__, __OS_BUILD_DATE__)

  Macro for displaying kernel build info: build time and build date.
- #define panic(fmt, ...) __panic(__FILE__, __func__, __LINE__, fmt, ##__VA_ARGS__)

  Macro for kernel panic (detecting an internal fatal error).

Functions

- void __panic (const char *file, const char *func, u32 line, const char *fmt,...)

  Detailed kernel panic function.
- void printk (const char *fmt,...)

  Formats and prints data.
- void vprintk (const char *fmt, va_list args)

  Auxilar function for formating and printing data.
- void kboot (multiboot_t *boot_info)

  Setup kernel.
- void kmain (u32 magic, multiboot_t *boot_info)

  Kernel entry point.

4.21.1 Detailed Description

Contains declarations for kernel functions and structures.

This header file includes constants, definitions related to the OS information, kernel setup, kernel entry point and several auxilary functions.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.21.2 Macro Definition Documentation

4.21.2.1 __DISPLAY_OS_BUILD_INFO

```
#define __DISPLAY_OS_BUILD_INFO( ) printk(__OS_BUILD_INFO_FMT__, " Build time:", __OS_BUILD_TIME__,
__OS_BUILD_DATE__)
```

Macro for displaying kernel build info: build time and build date.

4.21.2.2 __DISPLAY_OS_INFO

```
#define __DISPLAY_OS_INFO( ) printk(__OS_INFO_FMT__, __OS_NAME__, __OS_VERSION__, __OS_ARCH__)
```

Macro for displaying main OS info: name, version and architecture.

4.21.2.3 __OS_ARCH__

```
#define __OS_ARCH__ "x86_32"
```

4.21.2.4 __OS_BUILD_DATE__

```
#define __OS_BUILD_DATE__ __DATE__
```

4.21.2.5 __OS_BUILD_INFO_FMT__

```
#define __OS_BUILD_INFO_FMT__ "%s %s <%s>\n"
```

4.21.2.6 __OS_BUILD_TIME__

```
#define __OS_BUILD_TIME__ __TIME__
```

4.21.2.7 __OS_INFO_FMT__

```
#define __OS_INFO_FMT__ " %s (%s %s) (c) @alkuzin - 2024\n"
```

4.21.2.8 __OS_NAME__

```
#define __OS_NAME__ "nos"
```

OS information definitions.

4.21.2.9 __OS_VERSION__

```
#define __OS_VERSION__ "v0.0.2"
```

4.21.2.10 panic

```
#define panic(
              fmt,
              ... ) __panic(__FILE__, __func__, __LINE__, fmt, ##__VA_ARGS__)
```

Macro for kernel panic (detecting an internal fatal error).

It displaying error message, file, function and line where error occured.

Parameters

in	*fmt*	- given format string.
in	*...*	- given variable number of arguments.

4.21.3 Function Documentation

4.21.3.1 __panic()

```
void __panic (
            const char * file,
            const char * func,
            u32 line,
            const char * fmt,
             ...  )
```

Detailed kernel panic function.

It displaying error message, file, function and line where error occured.

Parameters

in	*file*	- given filename, where error occured.
in	*func*	- given function name, where error occured.
in	*line*	- given line number, where error occured.
in	*fmt*	- given format string.
in	*...*	- given variable number of arguments.

4.21.3.2 kboot()

```
void kboot (
            multiboot_t * boot_info )
```

Setup kernel.

Initializes kernel components such as TTY, GDT, IDT, timer and etc.

Parameters

in	*boot_info*	- given multiboot information structure.

4.21.3.3 kmain()

```
void kmain (
            u32 magic,
            multiboot_t * boot_info )  [extern]
```

Kernel entry point.

Parameters

in	*magic*	- given magic number.
in	*boot_info*	- given multiboot information structure.

4.21.3.4 printk()

```
void printk (
             const char * fmt,
              ...  )
```

Formats and prints data.

Parameters

in	*fmt*	- given format string.
in	*...*	- given variable number of arguments.

4.21.3.5 vprintk()

```
void vprintk (
             const char * fmt,
             va_list args )
```

Auxilar function for formating and printing data.

Parameters

in	*fmt*	- given format string.
in	*args*	- given variable number of arguments.

4.22 kernel.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

```
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_H_
00035 #define _NOS_KERNEL_H_
00036
00037 #include <stdint.h>
00038
00039 #include <nos/multiboot.h>
00040 #include <nos/tty.h>
00041
00043 #define __OS_NAME__             "nos"
00044 #define __OS_VERSION__          "v0.0.2"
00045 #define __OS_ARCH__             "x86_32"
00046 #define __OS_BUILD_DATE__       __DATE__
00047 #define __OS_BUILD_TIME__       __TIME__
00048 #define __OS_INFO_FMT__         " %s (%s %s) (c) @alkuzin - 2024\n"
00049 #define __OS_BUILD_INFO_FMT__   "%s %s <%s>\n"
00050
00055 #define __DISPLAY_OS_INFO() \
00056 printk(__OS_INFO_FMT__, __OS_NAME__, __OS_VERSION__, __OS_ARCH__)
00057
00062 #define __DISPLAY_OS_BUILD_INFO() \
00063 printk(__OS_BUILD_INFO_FMT__, " Build time:", __OS_BUILD_TIME__, __OS_BUILD_DATE__)
00064
00073 #define panic(fmt, ...) __panic(__FILE__, __func__, __LINE__, fmt, ##__VA_ARGS__)
00074
00075 /* detatiled kernel panic */
00087 void __panic(const char *file, const char *func, u32 line, const char *fmt, ...);
00088
00095 void printk(const char *fmt, ...);
00096
00103 void vprintk(const char *fmt, va_list args);
00104
00105 /* boot kernel */
00106
00115 void kboot(multiboot_t *boot_info);
00116
00123 extern void kmain(u32 magic, multiboot_t *boot_info);
00124
00125 #endif /* _NOS_KERNEL_H_ */
```

4.23 keyboard.h File Reference

Contains declarations for keyboard handeling functions and structures.

```
#include <stdint.h>
#include <nos/ports.h>
#include <nos/irq.h>
```

Macros

- #define INPUT_BUFFER_SIZE 256

Enumerations

- enum keycode_t {
  KEY_ESC = 0x01 , KEY_BACKSPACE = 0x0E , KEY_TAB = 0x0F , KEY_ENTER = 0x1C ,
  KEY_LCTRL = 0x1D , KEY_LSHFT = 0x2A , KEY_BACKSLASH = 0x2B , KEY_LALT = 0x38 ,
  KEY_SPACE = 0x39 , KEY_CAPS_LOCK = 0x3A , KEY_LEFT_ARROW = 0x4B , KEY_RIGHT_ARROW = 0x4D ,
  KEY_UP_ARROW = 0x48 , KEY_DOWN_ARROW = 0x50 }

  Keyboard special keys enumeration.

Functions

- void keyboard_init (void)

  keyboard initialization
- void keyboard_handler (int_reg_t *regs)

  Keyboard key press handler.
- void keyboard_wait (void)

  Keyboard wait for user to press a key.
- u8 keyboard_getchar (void)

  Keyboard get character on key press.

4.23.1 Detailed Description

Contains declarations for keyboard handeling functions and structures.

This header file includes definitions and functions related to the managing of keyboard interrupts.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.23.2 Macro Definition Documentation

4.23.2.1 INPUT_BUFFER_SIZE

```
#define INPUT_BUFFER_SIZE 256
```

4.23.3 Enumeration Type Documentation

4.23.3.1 keycode_t

```
enum keycode_t
```

Keyboard special keys enumeration.

Enumerator

KEY_ESC	
KEY_BACKSPACE	
KEY_TAB	
KEY_ENTER	
KEY_LCTRL	
KEY_LSHFT	
KEY_BACKSLASH	
KEY_LALT	
KEY_SPACE	
KEY_CAPS_LOCK	
KEY_LEFT_ARROW	
KEY_RIGHT_ARROW	

4.23.4 Function Documentation

4.23.4.1 keyboard_getchar()

```
u8 keyboard_getchar (
            void  )
```

Keyboard get character on key press.

Returns

Character read from the keyboard.

4.23.4.2 keyboard_handler()

```
void keyboard_handler (
            int_reg_t * regs )
```

Keyboard key press handler.

Parameters

in	*regs*	- given pointer to interrupt register state.

4.23.4.3 keyboard_init()

```
void keyboard_init (
            void  )
```

keyboard initialization

4.23.4.4 keyboard_wait()

```
void keyboard_wait (
            void  )
```

Keyboard wait for user to press a key.

4.24 keyboard.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

```
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_KEYBOARD_H_
00035 #define _NOS_KERNEL_KEYBOARD_H_
00036
00037 #include <stdint.h>
00038
00039 #include <nos/ports.h>
00040 #include <nos/irq.h>
00041
00042 #define INPUT_BUFFER_SIZE 256
00043
00045 typedef enum {
00046     KEY_ESC         = 0x01,
00047     KEY_BACKSPACE   = 0x0E,
00048     KEY_TAB         = 0x0F,
00049     KEY_ENTER       = 0x1C,
00050     KEY_LCTRL       = 0x1D,
00051     KEY_LSHFT       = 0x2A,
00052     KEY_BACKSLASH   = 0x2B,
00053     KEY_LALT        = 0x38,
00054     KEY_SPACE       = 0x39,
00055     KEY_CAPS_LOCK   = 0x3A,
00056     KEY_LEFT_ARROW  = 0x4B,
00057     KEY_RIGHT_ARROW = 0x4D,
00058     KEY_UP_ARROW    = 0x48,
00059     KEY_DOWN_ARROW  = 0x50
00060 } keycode_t;
00061
00063 void keyboard_init(void);
00064
00070 void keyboard_handler(int_reg_t *regs);
00071
00073 void keyboard_wait(void);
00074
00080 u8   keyboard_getchar(void);
00081
00082 #endif /* _NOS_KERNEL_KEYBOARD_H_ */
```

4.25 kmalloc.h File Reference

Contains declarations for dynamic heap allocation management.

```
#include <stdint.h>
#include <stddef.h>
#include <nos/pmm.h>
#include <nos/vmm.h>
```

Data Structures

- struct kmalloc_block_s

  Structure representing a block of memory for kernel dynamic memory allocation.

Macros

- #define PAGE_SIZE 4096

Typedefs

- typedef struct kmalloc_block_s kmalloc_block_t

  Structure representing a block of memory for kernel dynamic memory allocation.

Functions

- void * kmalloc_get_head (void)

  Get the start of the kmalloc blocks linked list.
- void kmalloc_init (const usize n)

  Initialize kernel dynamic memory allocation.
- void kmalloc_split (kmalloc_block_t *node, const u32 size)

  Split a memory block into two by inserting a new block.
- void * kmalloc_next_block (const u32 size)

  Find and allocate the next block of memory.
- void kmalloc_merge_free_blocks (void)

  Merge free blocks of memory to partially prevent memory fragmentation.
- void kmalloc_free (void *ptr)

  Free allocated memory.

4.25.1 Detailed Description

Contains declarations for dynamic heap allocation management.

This header file includes definitions and functions related to the managing dynamic heap allocation.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.25.2 Macro Definition Documentation

4.25.2.1 PAGE_SIZE

```
#define PAGE_SIZE 4096
```

4.25.3 Typedef Documentation

4.25.3.1 kmalloc_block_t

```
typedef struct kmalloc_block_s kmalloc_block_t
```

Structure representing a block of memory for kernel dynamic memory allocation.

4.25.4 Function Documentation

4.25.4.1 kmalloc_free()

```
void kmalloc_free (
            void * ptr )
```

Free allocated memory.

Parameters

in	*ptr*	- given pointer to the memory block to free.

4.25.4.2 kmalloc_get_head()

```
void * kmalloc_get_head (
            void  )
```

Get the start of the kmalloc blocks linked list.

Returns

Pointer to the head of the kmalloc blocks linked list.

4.25.4.3 kmalloc_init()

```
void kmalloc_init (
            const usize n )
```

Initialize kernel dynamic memory allocation.

Parameters

in	*n*	- given size of memory to initialize.

4.25.4.4 kmalloc_merge_free_blocks()

```
void kmalloc_merge_free_blocks (
            void  )
```

Merge free blocks of memory to partially prevent memory fragmentation.

4.25.4.5 kmalloc_next_block()

```
void * kmalloc_next_block (
            const u32 size )
```

Find and allocate the next block of memory.

Parameters

in	*size*	- given dize of the memory block to allocate.

Returns

Pointer to the allocated memory block.

4.25.4.6 kmalloc_split()

```
void kmalloc_split (
            kmalloc_block_t * node,
            const u32 size )
```

Split a memory block into two by inserting a new block.

Parameters

in	*node*	- given pointer to the memory block to split.
in	*size*	- given size of the new block to insert.

4.26 kmalloc.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_KMALLOC_H_
00035 #define _NOS_KERNEL_KMALLOC_H_
00036
00037 #include <stdint.h>
00038 #include <stddef.h>
00039
00040 #include <nos/pmm.h>
00041 #include <nos/vmm.h>
00042
00043 #define PAGE_SIZE 4096
00044
00046 typedef struct kmalloc_block_s {
00047     usize size;
00048     bool is_free;
00049     struct kmalloc_block_s *next;
00050 } kmalloc_block_t;
00051
00057 void *kmalloc_get_head(void);
00058
00064 void kmalloc_init(const usize n);
00065
00072 void kmalloc_split(kmalloc_block_t *node, const u32 size);
00073
00080 void *kmalloc_next_block(const u32 size);
00081
00083 void kmalloc_merge_free_blocks(void);
```

```
00084
00090 void kmalloc_free(void *ptr);
00091
00092 #endif /* _NOS_KERNEL_KMALLOC_H_ */
```

4.27 mm.h File Reference

Contains declarations for memory management.

```
#include <nos/multiboot.h>
#include <nos/pmm.h>
#include <nos/vmm.h>
```

Functions

- void memory_init (multiboot_t *boot_info)

  Initialize memory manager.

4.27.1 Detailed Description

Contains declarations for memory management.

This header file includes functions related to the physical and virtual memory management.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.27.2 Function Documentation

4.27.2.1 memory_init()

```
void memory_init (
            multiboot_t * boot_info )
```

Initialize memory manager.

Parameters

in	*boot_info*	- given multiboot information structure.

4.28 mm.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_MM_H_
00035 #define _NOS_KERNEL_MM_H_
00036
00037 #include <nos/multiboot.h>
00038 #include <nos/pmm.h>
00039 #include <nos/vmm.h>
00040
00046 void memory_init(multiboot_t *boot_info);
00047
00048 #endif /* _NOS_KERNEL_MM_H_ */
```

4.29 multiboot.h File Reference

Contains multiboot information structures decalarations.

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

Data Structures

- struct multiboot_aout_symbol_table_s

  Structure representing the symbol table for a.out format.
- struct multiboot_elf_section_header_table_s

  Structure representing the section header table for ELF format.
- struct multiboot_t

  Type representing multiboot information.
- struct multiboot_mmap_entry_s

  Structure representing a memory map entry in multiboot format.

Macros

- #define MULTIBOOT_MEMORY_AVAILABLE 1
- #define MULTIBOOT_MEMORY_RESERVED 2
- #define MULTIBOOT_MEMORY_ACPI_RECLAIMABLE 3
- #define MULTIBOOT_MEMORY_NVS 4
- #define MULTIBOOT_MEMORY_BADRAM 5

Typedefs

- typedef struct multiboot_mmap_entry_s multiboot_mmap_entry_t
  Type representing a memory map entry in multiboot format.

Functions

- struct multiboot_mmap_entry_s __attribute__ ((packed))

Variables

- u32 size
  Size of the memory map entry.
- u32 addr_low
  Lower address of the memory region.
- u32 addr_high
  Higher address of the memory region.
- u32 len_low
  Lower length of the memory region.
- u32 len_high
  Higher length of the memory region.
- u32 type
  Type of memory region.

4.29.1 Detailed Description

Contains multiboot information structures decalarations.

This header file includes functions related to the physical and virtual memory managing.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.29.2 Macro Definition Documentation

4.29.2.1 MULTIBOOT_MEMORY_ACPI_RECLAIMABLE

```
#define MULTIBOOT_MEMORY_ACPI_RECLAIMABLE 3
```

4.29.2.2 MULTIBOOT_MEMORY_AVAILABLE

```
#define MULTIBOOT_MEMORY_AVAILABLE 1
```

4.29.2.3 MULTIBOOT_MEMORY_BADRAM

```
#define MULTIBOOT_MEMORY_BADRAM 5
```

4.29.2.4 MULTIBOOT_MEMORY_NVS

```
#define MULTIBOOT_MEMORY_NVS 4
```

4.29.2.5 MULTIBOOT_MEMORY_RESERVED

```
#define MULTIBOOT_MEMORY_RESERVED 2
```

4.29.3 Typedef Documentation

4.29.3.1 multiboot_mmap_entry_t

```
typedef struct multiboot_mmap_entry_s multiboot_mmap_entry_t
```

Type representing a memory map entry in multiboot format.

4.29.4 Function Documentation

4.29.4.1 __attribute__()

```
struct multiboot_mmap_entry_s __attribute__ (
            (packed)  )
```

4.29.5 Variable Documentation

4.29.5.1 addr_high

```
u32 addr_high
```

Higher address of the memory region.

4.29.5.2 addr_low

```
u32 addr_low
```

Lower address of the memory region.

4.29.5.3 len_high

```
u32 len_high
```

Higher length of the memory region.

4.29.5.4 len_low

```
u32 len_low
```

Lower length of the memory region.

4.29.5.5 size

```
u32 size
```

Size of the memory map entry.

4.29.5.6 type

```
u32 type
```

Type of memory region.

4.30 multiboot.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_MULTIBOOT_H_
00035 #define _NOS_KERNEL_MULTIBOOT_H_
00036
00037 #include <stdint.h>
00038
00040 struct multiboot_aout_symbol_table_s {
00041     u32 tabsize;
00042     u32 strsize;
00043     u32 addr;
00044     u32 reserved;
00045 };
00046
00048 struct multiboot_elf_section_header_table_s {
```

```
00049     u32 num;
00050     u32 size;
00051     u32 addr;
00052     u32 shndx;
00053 };
00054
00056 typedef struct {
00057     u32 flags;
00058     u32 mem_lower;
00059     u32 mem_upper;
00060     u32 boot_device;
00061     u32 cmdline;
00062     u32 mods_count;
00063     u32 mods_addr;
00064     union {
00065         struct multiboot_aout_symbol_table_s aout_sym;
00066         struct multiboot_elf_section_header_table_s elf_sec;
00067     } u;
00068     u32 mmap_length;
00069     u32 mmap_addr;
00070     u32 drives_length;
00071     u32 drives_addr;
00072     u32 config_table;
00073     u32 boot_loader_name;
00074     u32 apm_table;
00075     u32 vbe_control_info;
00076     u32 vbe_mode_info;
00077     u16 vbe_mode;
00078     u16 vbe_interface_seg;
00079     u16 vbe_interface_off;
00080     u16 vbe_interface_len;
00081 } multiboot_t;
00082
00084 struct multiboot_mmap_entry_s {
00085     u32 size;
00086     u32 addr_low;
00087     u32 addr_high;
00088     u32 len_low;
00089     u32 len_high;
00090 #define MULTIBOOT_MEMORY_AVAILABLE              1
00091 #define MULTIBOOT_MEMORY_RESERVED               2
00092 #define MULTIBOOT_MEMORY_ACPI_RECLAIMABLE       3
00093 #define MULTIBOOT_MEMORY_NVS                    4
00094 #define MULTIBOOT_MEMORY_BADRAM                 5
00095     u32 type;
00096 } __attribute__((packed));
00097
00099 typedef struct multiboot_mmap_entry_s multiboot_mmap_entry_t;
00100
00101 #endif /* _NOS_KERNEL_MULTIBOOT_H_ */
```

4.31 pmm.h File Reference

Contains declarations for physical memory management.

```
#include <stdint.h>
#include <stddef.h>
#include <nos/multiboot.h>
```

Macros

- #define BLOCK_SIZE 4096 /* 4KB */
- #define BITS_PER_BYTE 8

Kernel end address.

Functions

- void pmm_set_block (u32 bit)
  Set block in the memory map.
- void pmm_unset_block (u32 bit)
  Unset block in the memory map.
- bool pmm_test_block (u32 bit)
  Test if a block in the memory map is set/used.
- i32 pmm_find_first_free_blocks (u32 n)
  Find the first free blocks in the memory map.
- void pmm_init (u32 start_addr, u32 size)
  Initialize the physical memory manager.
- void pmm_get_memory (const multiboot_t *boot_info, u32 *start_addr, u32 *size)
  Get information about memory regions. Get largest free area of RAM & get free and total physical memory.
- void pmm_region_init (u32 base_addr, u32 size)
  Initialize a memory region.
- void pmm_region_deinit (u32 base_addr, u32 size)
  Deinitialize a memory region.
- u32 * pmm_blocks_alloc (u32 n)
  Allocate a block of memory.
- void pmm_free_blocks (u32 *addr, u32 n)
  Free a block of memory.
- void pmm_display_memory (multiboot_t *boot_info)
  Display memory information.

Variables

- u32 _kernel_end

4.31.1 Detailed Description

Contains declarations for physical memory management.

This header file includes functions related to the physical memory management.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.31.2 Macro Definition Documentation

4.31.2.1 BITS_PER_BYTE

```
#define BITS_PER_BYTE 8
```

Kernel end address.

4.31.2.2 BLOCK_SIZE

```
#define BLOCK_SIZE 4096 /* 4KB */
```

4.31.3 Function Documentation

4.31.3.1 pmm_blocks_alloc()

```
u32 * pmm_blocks_alloc (
            u32 n )
```

Allocate a block of memory.

Parameters

in	*n*	- given number of blocks to allocate.

Returns

pointer to the allocated memory block.

4.31.3.2 pmm_display_memory()

```
void pmm_display_memory (
            multiboot_t * boot_info )
```

Display memory information.

Parameters

in	*boot_info*	- given multiboot information structure.

4.31.3.3 pmm_find_first_free_blocks()

```
i32 pmm_find_first_free_blocks (
            u32 n )
```

Find the first free blocks in the memory map.

Parameters

in	*n*	- given number of blocks to find.

Returns

the index of the first free block, or -1 if not found.

4.31.3.4 pmm_free_blocks()

```
void pmm_free_blocks (
            u32 * addr,
            u32 n )
```

Free a block of memory.

Parameters

in	*addr*	- given pointer to the memory block to free.
in	*n*	- given number of blocks to free.

4.31.3.5 pmm_get_memory()

```
void pmm_get_memory (
            const multiboot_t * boot_info,
            u32 * start_addr,
            u32 * size )
```

Get information about memory regions. Get largest free area of RAM & get free and total physical memory.

Parameters

in	*boot_info*	- given multiboot information structure.
in	*start_addr*	- given starting address of memory.
in	*size*	- given pointer to store the size of memory.

4.31.3.6 pmm_init()

```
void pmm_init (
            u32 start_addr,
            u32 size )
```

Initialize the physical memory manager.

Parameters

in	*start_addr*	- given starting address of the memory map.
in	*size*	- given size of the memory map.

4.31.3.7 pmm_region_deinit()

```
void pmm_region_deinit (
            u32 base_addr,
            u32 size )
```

Deinitialize a memory region.

Parameters

in	*base_addr*	- given base address of the region.
in	*size*	- given size of the region.

4.31.3.8 pmm_region_init()

```
void pmm_region_init (
             u32 base_addr,
             u32 size )
```

Initialize a memory region.

Parameters

in	*base_addr*	- given base address of the region.
in	*size*	- given size of the region.

4.31.3.9 pmm_set_block()

```
void pmm_set_block (
             u32 bit )
```

Set block in the memory map.

Parameters

in	*bit*	- given block to set.

4.31.3.10 pmm_test_block()

```
bool pmm_test_block (
             u32 bit )
```

Test if a block in the memory map is set/used.

Parameters

in	*bit*	- given block to test.

Returns

true - block is set;

false - otherwise.

4.31.3.11 pmm_unset_block()

```
void pmm_unset_block (
            u32 bit )
```

Unset block in the memory map.

Parameters

in	*bit*	- given block to unset.

4.31.4 Variable Documentation

4.31.4.1 _kernel_end

```
u32 _kernel_end  [extern]
```

4.32 pmm.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_PMM_H_
00035 #define _NOS_KERNEL_PMM_H_
00036
00037 #include <stdint.h>
00038 #include <stddef.h>
00039
00040 #include <nos/multiboot.h>
00041
00042 #define BLOCK_SIZE     4096 /* 4KB */
00043 #define BITS_PER_BYTE  8
00044
00046 extern u32 _kernel_end;
00047
00053 void pmm_set_block(u32 bit);
00054
00060 void pmm_unset_block(u32 bit);
00061
00069 bool pmm_test_block(u32 bit);
00070
00077 i32 pmm_find_first_free_blocks(u32 n);
00078
00085 void pmm_init(u32 start_addr, u32 size);
00086
00095 void pmm_get_memory(const multiboot_t *boot_info, u32 *start_addr, u32 *size);
00096
```

```
00103 void pmm_region_init(u32 base_addr, u32 size);
00104
00111 void pmm_region_deinit(u32 base_addr, u32 size);
00112
00119 u32 *pmm_blocks_alloc(u32 n);
00120
00127 void pmm_free_blocks(u32 *addr, u32 n);
00128
00134 void pmm_display_memory(multiboot_t *boot_info);
00135
00136 #endif /* _NOS_KERNEL_PMM_H_ */
```

4.33 ports.h File Reference

Contains functions for input/output operations on ports.

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

Functions

- static void outb (u16 port, u8 data)

  Output a byte to a specified port.
- static u8 inb (u16 port)

  Receive a byte of data from a specified input/output port.

4.33.1 Detailed Description

Contains functions for input/output operations on ports.

This header file contains inline functions for input/output operations on ports in x86 architecture.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.33.2 Function Documentation

4.33.2.1 inb()

```
static u8 inb (
            u16 port )  [inline], [static]
```

Receive a byte of data from a specified input/output port.

Parameters

port	- given port from which the data will be read.

Returns

the byte of data read from the port.

4.33.2.2 outb()

```
static void outb (
             u16 port,
             u8 data )  [inline], [static]
```

Output a byte to a specified port.

Parameters

in	*port*	-given port to which the data will be written.
in	*data*	- given data byte to be written to the port.

4.34 ports.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00035 #ifndef _NOS_KERNEL_PORTS_H_
00036 #define _NOS_KERNEL_PORTS_H_
00037
00038 #include <stdint.h>
00039
00046 static inline void outb(u16 port, u8 data)
00047 {
00048     __asm__ volatile("outb %1, %0" : : "dN" (port), "a" (data));
00049 }
00050
00057 static inline u8 inb(u16 port)
00058 {
00059     u8 rv;
00060     __asm__ volatile("inb %1, %0" : "=a" (rv) : "dN" (port));
00061     return rv;
00062 }
00063
00064 #endif /* _NOS_KERNEL_PORTS_H_ */
```

4.35 ksh.h File Reference

Contain kernel shell functions.

```
#include <stddef.h>
#include <stdint.h>
#include <nos/multiboot.h>
```

Enumerations

- enum theme_t { THEME_DEFAULT , THEME_CLASSIC }
  Enumeration of builtin CLI themes.

Functions

- void ksh_init (multiboot_t *boot_info)
  Initialize kernel shell.
- void ksh_exec (multiboot_t *boot_info, const char *cmd)
  Execute builtin shell commands.
- void ksh_warning (const char *cmd)
  print shell warning in case of incorrect command.
- void ksh_clear (void)
  Clear terminal.
- void ksh_lsmem (multiboot_t *boot_info)
  Display list of available memory.
- void ksh_help (void)
  Display list of available shell commands.
- void ksh_theme (theme_t theme)
  Change CLI theme.

4.35.1 Detailed Description

Contain kernel shell functions.

This header file contain main kernel shell functions and builtin commands.

Author

Alexander Kuzin (alkuzin)

Date

15.05.2024

4.35.2 Enumeration Type Documentation

4.35.2.1 theme_t

```
enum theme_t
```

Enumeration of builtin CLI themes.

Enumerator

THEME_DEFAULT	
THEME_CLASSIC	

4.35.3 Function Documentation

4.35.3.1 ksh_clear()

```
void ksh_clear (
            void  )
```

Clear terminal.

4.35.3.2 ksh_exec()

```
void ksh_exec (
            multiboot_t * boot_info,
            const char * cmd )
```

Execute builtin shell commands.

Parameters

in	*boot_info*	- given multiboot information structure.
in	*cmd*	- given shell command string.

4.35.3.3 ksh_help()

```
void ksh_help (
            void  )
```

Display list of available shell commands.

4.35.3.4 ksh_init()

```
void ksh_init (
            multiboot_t * boot_info )
```

Initialize kernel shell.

Parameters

in	*boot_info*	- given multiboot information structure.

4.35.3.5 ksh_lsmem()

```
void ksh_lsmem (
            multiboot_t * boot_info )
```

Display list of available memory.

Parameters

in	*boot_info*	- given multiboot information structure.

4.35.3.6 ksh_theme()

```
void ksh_theme (
            theme_t theme )
```

Change CLI theme.

Parameters

in	*theme*	- given theme type.

4.35.3.7 ksh_warning()

```
void ksh_warning (
            const char * cmd )
```

print shell warning in case of incorrect command.

Parameters

in	*cmd*	- given shell command string.

4.36 ksh.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
```

```
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_SHELL_H_
00035 #define _NOS_KERNEL_SHELL_H_
00036
00037 #include <stddef.h>
00038 #include <stdint.h>
00039
00040 #include <nos/multiboot.h>
00041
00043 typedef enum {
00044     THEME_DEFAULT,
00045     THEME_CLASSIC
00046 } theme_t;
00047
00053 void ksh_init(multiboot_t *boot_info);
00054
00061 void ksh_exec(multiboot_t *boot_info, const char *cmd);
00062
00068 void ksh_warning(const char *cmd);
00069
00071 void ksh_clear(void);
00072
00078 void ksh_lsmem(multiboot_t *boot_info);
00079
00081 void ksh_help(void);
00082
00088 void ksh_theme(theme_t theme);
00089
00090 #endif /* _NOS_KERNEL_SHELL_H_ */
```

4.37 timer.h File Reference

Contains functions related to timer operations.

`#include <nos/irq.h>`

Functions

- void timer_init (void)

  Initialize the timer.
- void on_irq0 (int_reg_t *regs)

  Interrupt service routine for IRQ0.

4.37.1 Detailed Description

Contains functions related to timer operations.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.37.2 Function Documentation

4.37.2.1 on_irq0()

```
void on_irq0 (
             int_reg_t * regs )
```

Interrupt service routine for IRQ0.

4.37.2.2 timer_init()

```
void timer_init (
             void  )
```

Initialize the timer.

4.38 timer.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00031 #ifndef _NOS_KERNEL_TIMER_H_
00032 #define _NOS_KERNEL_TIMER_H_
00033
00034 #include <nos/irq.h>
00035
00036
00038 void timer_init(void);
00039
00041 void on_irq0(int_reg_t *regs);
00042
00043 #endif /* _NOS_KERNEL_TIMER_H_ */
```

4.39 tty.h File Reference

TTY (teletype terminal). Contains definitions related to screen input/output.

```
#include <stdarg.h>
#include <nos/vga.h>
```

Data Structures

- struct tty_s

Macros

- #define TTY_FG_COLOR VGA_COLOR_WHITE

  < *Default kernel TTY foreground & background color.*
- #define TTY_BG_COLOR VGA_COLOR_BLUE
- #define TTY_TAB_WIDTH 4

  For NULL pointer in kprintf.
- #define __NIL__ "(nil)"

  TTY management structure.

Typedefs

- typedef struct tty_s tty_t

  initialize kernel TTY structure.

Functions

- void tty_init (void)
- i32 tty_get_x (void)

  Get cursor x position.
- i32 tty_get_y (void)

  Get cursor y position.
- void tty_set_x (i32 x)

  Set cursor x position.
- void tty_set_y (i32 y)

  Set cursor y position.
- vga_color_t tty_get_fg (void)

  Get kernel TTY structure foreground color.
- vga_color_t tty_get_bg (void)

  Get kernel TTY structure background color.
- void tty_set_color (vga_color_t fg, vga_color_t bg)

  Set foreground & background color.
- i32 tty_get_height (void)

  Get screen height.
- i32 tty_get_width (void)

  Get screen width.
- void tty_clear (void)

  Rewrite TTY buffer.
- void tty_rewrite (void)
- void tty_kputchar_at (char c, u8 color, i32 x, i32 y)

  Print char with custom color in a specific place.
- void kputchar (const i32 c)

  Print character to screen.

4.39.1 Detailed Description

TTY (teletype terminal). Contains definitions related to screen input/output.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.39.2 Macro Definition Documentation

4.39.2.1 __NIL__

```
#define __NIL__ "(nil)"
```

TTY management structure.

4.39.2.2 TTY_BG_COLOR

```
#define TTY_BG_COLOR VGA_COLOR_BLUE
```

4.39.2.3 TTY_FG_COLOR

```
#define TTY_FG_COLOR VGA_COLOR_WHITE
```

< Default kernel TTY foreground & background color.

4.39.2.4 TTY_TAB_WIDTH

```
#define TTY_TAB_WIDTH 4
```

For NULL pointer in kprintf.

4.39.3 Typedef Documentation

4.39.3.1 tty_t

```
typedef struct tty_s tty_t
```

initialize kernel TTY structure.

4.39.4 Function Documentation

4.39.4.1 kputchar()

```
void kputchar (
            const i32 c )
```

Print character to screen.

Parameters

in	*c*	- given character to print.

4.39.4.2 tty_clear()

```
void tty_clear (
            void  )
```

Rewrite TTY buffer.

4.39.4.3 tty_get_bg()

```
vga_color_t tty_get_bg (
            void  )
```

Get kernel TTY structure background color.

Returns

current background color.

4.39.4.4 tty_get_fg()

```
vga_color_t tty_get_fg (
            void  )
```

Get kernel TTY structure foreground color.

Returns

current foreground color.

4.39.4.5 tty_get_height()

```
i32 tty_get_height (
            void  )
```

Get screen height.

Returns

current screen height.

4.39.4.6 tty_get_width()

```
i32 tty_get_width (
            void  )
```

Get screen width.

Returns

current screen width. Clear screen.

4.39.4.7 tty_get_x()

```
i32 tty_get_x (
            void  )
```

Get cursor x position.

Returns

x position.

4.39.4.8 tty_get_y()

```
i32 tty_get_y (
            void  )
```

Get cursor y position.

Returns

y position.

4.39.4.9 tty_init()

```
void tty_init (
            void  )
```

4.39.4.10 tty_kputchar_at()

```
void tty_kputchar_at (
            char c,
            u8 color,
            i32 x,
            i32 y )
```

Print char with custom color in a specific place.

Parameters

in	*c*	- given character to print.
in	*color*	- given custom color.
in	*x*	- given cursor x position to print.
in	*y*	- given cursor y position to print.

4.39.4.11 tty_rewrite()

```
void tty_rewrite (
            void  )
```

4.39.4.12 tty_set_color()

```
void tty_set_color (
            vga_color_t fg,
            vga_color_t bg )
```

Set foreground & background color.

Parameters

in	*fg*	- given foreground color.
in	*bg*	- given background color.

4.39.4.13 tty_set_x()

```
void tty_set_x (
            i32 x )
```

Set cursor x position.

Parameters

in	*x*	- new given cursor x position.

4.39.4.14 tty_set_y()

```
void tty_set_y (
            i32 y )
```

Set cursor y position.

Parameters

in	*y*	- new given cursor y position.

4.40 tty.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00032 #ifndef _NOS_KERNEL_TTY_H_
00033 #define _NOS_KERNEL_TTY_H_
00034
00035 #include <stdarg.h>
00036
00037 #include <nos/vga.h>
00038
00040 #define TTY_FG_COLOR  VGA_COLOR_WHITE
00041 #define TTY_BG_COLOR  VGA_COLOR_BLUE
00042 #define TTY_TAB_WIDTH 4
00043
00045 #define __NIL__ "(nil)"
00046
00048 typedef struct tty_s {
00049     u16 *v_mem;
00050     i32 x_pos;
00051     i32 y_pos;
00052     vga_color_t fg;
00053     vga_color_t bg;
00054     u8  color;
00055     i32 height;
00056     i32 width;
00057 } tty_t;
00058
00060 void tty_init(void);
00061
00067 i32  tty_get_x(void);
00068
00074 i32  tty_get_y(void);
00075
00081 void tty_set_x(i32 x);
00082
00088 void tty_set_y(i32 y);
00089
00095 vga_color_t tty_get_fg(void);
00096
00102 vga_color_t tty_get_bg(void);
00103
00110 void tty_set_color(vga_color_t fg, vga_color_t bg);
00111
00117 i32  tty_get_height(void);
00118
00124 i32  tty_get_width(void);
00125
00127 void tty_clear(void);
00128
00130 void tty_rewrite(void);
00131
00140 void tty_kputchar_at(char c, u8 color, i32 x, i32 y);
00141
00147 void kputchar(const i32 c);
00148
00149 #endif /* _NOS_KERNEL_TTY_H_ */
```

4.41 libc/unistd.h File Reference

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

Macros

- #define stdin 0
- #define stdout 1
- #define stderr 2

Functions

- usize write (i32 fd, const void *buffer, usize count)

4.41.1 Macro Definition Documentation

4.41.1.1 stderr

```
#define stderr 2
```

4.41.1.2 stdin

```
#define stdin 0
```

4.41.1.3 stdout

```
#define stdout 1
```

4.41.2 Function Documentation

4.41.2.1 write()

```
usize write (
            i32 fd,
            const void * buffer,
            usize count )
```

4.42 libc/unistd.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00023 #ifndef _LIBC_UNISTD_H_
00024 #define _LIBC_UNISTD_H_
00025
00026 #include <stdint.h>
00027
00028 // TODO: put this defines into the <stdio.h>
00029 #define stdin  0
00030 #define stdout 1
00031 #define stderr 2
00032
00033 /* writes up to count bytes from the buffer starting at 'buffer'
00034  * to the file referred to by the file descriptor 'fd' */
00035 usize write(i32 fd, const void *buffer, usize count);
00036
00037 #endif /* _LIBC_UNISTD_H_ */
```

4.43 nos/unistd.h File Reference

Contains diferent system functions.

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

Functions

- void __ksleep (u32 microsec)

  Kernel time delay in microseconds.
- void ksleep (u32 sec)

  Kernel time delay in seconds.
- void khalt (void)
- void * kmalloc (usize n)

  Allocates n bytes and returns a pointer to the allocated memory.
- void kfree (void *ptr)

  Frees the memory space pointed to by ptr, which must have been returned by a previous call to kmalloc() or related functions. Otherwise, or if ptr has already been freed, undefined behavior occurs. If ptr is null pointer, no operation is performed.

4.43.1 Detailed Description

Contains diferent system functions.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.43.2 Function Documentation

4.43.2.1 __ksleep()

```
void __ksleep (
            u32 microsec )
```

Kernel time delay in microseconds.

Parameters

in	*microsec*	- given number of microseconds for kernel to sleep.

4.43.2.2 kfree()

```
void kfree (
            void * ptr )
```

Frees the memory space pointed to by ptr, which must have been returned by a previous call to kmalloc() or related functions. Otherwise, or if ptr has already been freed, undefined behavior occurs. If ptr is null pointer, no operation is performed.

Parameters

in	*ptr*	- given pointer to allocated memory.

4.43.2.3 khalt()

```
void khalt (
            void  )
```

4.43.2.4 kmalloc()

```
void * kmalloc (
            usize n )
```

Allocates n bytes and returns a pointer to the allocated memory.

Parameters

in	*n*	- given number of bytes to allocate.

Returns

pointer to allocated memory in case of success.

null pointer otherwise.

4.43.2.5 ksleep()

```
void ksleep (
             u32 sec )
```

Kernel time delay in seconds.

Parameters

in	*sec*	- given number of seconds for kernel to sleep. Halt kernel.

4.44 nos/unistd.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00031 #ifndef _NOS_UNISTD_H_
00032 #define _NOS_UNISTD_H_
00033
00034 #include <stdint.h>
00035 #include <stddef.h>
00036
00042 void __ksleep(u32 microsec);
00043
00049 void ksleep(u32 sec);
00050
00052 void khalt(void);
00053
00062 void *kmalloc(usize n);
00063
00064
00074 void kfree(void *ptr);
00075
00076 #endif /* _NOS_UNISTD_H_ */
```

4.45 vga.h File Reference

Contains definitions related to screen characters input/output.

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

Macros

- #define VIDEO_MEMORY 0xB8000
  < *VGA screen information macros.*
- #define VGA_SCREEN_WIDTH 80
- #define VGA_SCREEN_HEIGHT 25
- #define REG_SCREEN_CTRL 0x3D4
- #define REG_SCREEN_DATA 0x3D5

  VGA colors enumeration.

Typedefs

- typedef enum vga_color vga_color_t

Enumerations

- enum vga_color {
  VGA_COLOR_BLACK , VGA_COLOR_BLUE , VGA_COLOR_GREEN , VGA_COLOR_CYAN ,
  VGA_COLOR_RED , VGA_COLOR_MAGENTA , VGA_COLOR_BROWN , VGA_COLOR_LIGHT_GREY ,
  VGA_COLOR_DARK_GREY , VGA_COLOR_LIGHT_BLUE , VGA_COLOR_LIGHT_GREEN , VGA_COLOR_LIGHT_CYAN
  ,
  VGA_COLOR_LIGHT_RED , VGA_COLOR_LIGHT_MAGENTA , VGA_COLOR_YELLOW , VGA_COLOR_WHITE
  }

Functions

- u8 vga_entry_color (vga_color_t fg, vga_color_t bg)

  Creates a VGA color entry based on the foreground and background colors.
- u16 vga_entry (u8 c, u8 color)

  Creates a VGA entry combining a character and color information.
- void update_cursor (i32 x, i32 y)

  Updates the cursor position on the screen.

4.45.1 Detailed Description

Contains definitions related to screen characters input/output.

VGA (Video Graphics Array). It is a standard for displaying graphics and video on computer monitors.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.45.2 Macro Definition Documentation

4.45.2.1 REG_SCREEN_CTRL

```
#define REG_SCREEN_CTRL 0x3D4
```

4.45.2.2 REG_SCREEN_DATA

```
#define REG_SCREEN_DATA 0x3D5
```

VGA colors enumeration.

4.45.2.3 VGA_SCREEN_HEIGHT

```
#define VGA_SCREEN_HEIGHT 25
```

4.45.2.4 VGA_SCREEN_WIDTH

```
#define VGA_SCREEN_WIDTH 80
```

4.45.2.5 VIDEO_MEMORY

```
#define VIDEO_MEMORY 0xB8000
```

< VGA screen information macros.

4.45.3 Typedef Documentation

4.45.3.1 vga_color_t

```
typedef enum vga_color vga_color_t
```

Enumerator

4.45.4 Enumeration Type Documentation

4.45.4.1 vga_color

```
enum vga_color
```

Enumerator

VGA_COLOR_BLACK	
VGA_COLOR_BLUE	
VGA_COLOR_GREEN	
VGA_COLOR_CYAN	
VGA_COLOR_RED	
VGA_COLOR_MAGENTA	
VGA_COLOR_BROWN	
VGA_COLOR_LIGHT_GREY	
VGA_COLOR_DARK_GREY	
VGA_COLOR_LIGHT_BLUE	
VGA_COLOR_LIGHT_GREEN	
VGA_COLOR_LIGHT_CYAN	
VGA_COLOR_LIGHT_RED	
VGA_COLOR_LIGHT_MAGENTA	
VGA_COLOR_YELLOW	
VGA_COLOR_WHITE	

4.45.5 Function Documentation

4.45.5.1 update_cursor()

```
void update_cursor (
            i32 x,
            i32 y )
```

Updates the cursor position on the screen.

Parameters

in	*x*	- given x position of the cursor.
in	*y*	- given y position of the cursor.

4.45.5.2 vga_entry()

```
u16 vga_entry (
            u8 c,
            u8 color )
```

Creates a VGA entry combining a character and color information.

Parameters

in	*c*	- given character.
in	*color*	- given color information for the character.

Returns

the VGA entry combining character and color.

4.45.5.3 vga_entry_color()

```
u8 vga_entry_color (
            vga_color_t fg,
            vga_color_t bg )
```

Creates a VGA color entry based on the foreground and background colors.

Parameters

in	*fg*	- given foreground color.
in	*bg*	- given background color.

Returns

the VGA color entry.

4.46 vga.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_VGA_H_
00035 #define _NOS_KERNEL_VGA_H_
00036
00037 #include <stdint.h>
00038
00040 #define VIDEO_MEMORY       0xB8000
```

```
00041 #define VGA_SCREEN_WIDTH   80
00042 #define VGA_SCREEN_HEIGHT  25
00043
00044 #define REG_SCREEN_CTRL    0x3D4
00045 #define REG_SCREEN_DATA    0x3D5
00046
00048 typedef enum vga_color {
00049     VGA_COLOR_BLACK,
00050     VGA_COLOR_BLUE,
00051     VGA_COLOR_GREEN,
00052     VGA_COLOR_CYAN,
00053     VGA_COLOR_RED,
00054     VGA_COLOR_MAGENTA,
00055     VGA_COLOR_BROWN,
00056     VGA_COLOR_LIGHT_GREY,
00057     VGA_COLOR_DARK_GREY,
00058     VGA_COLOR_LIGHT_BLUE,
00059     VGA_COLOR_LIGHT_GREEN,
00060     VGA_COLOR_LIGHT_CYAN,
00061     VGA_COLOR_LIGHT_RED,
00062     VGA_COLOR_LIGHT_MAGENTA,
00063     VGA_COLOR_YELLOW,
00064     VGA_COLOR_WHITE
00065 } vga_color_t;
00066
00075 u8   vga_entry_color(vga_color_t fg, vga_color_t bg);
00076
00085 u16  vga_entry(u8 c, u8 color);
00086
00093 void update_cursor(i32 x, i32 y);
00094
00095 #endif /* _NOS_KERNEL_VGA_H_ */
```

4.47 vmm.h File Reference

Contains declarations for virtual memory management.

```
#include <stdint.h>
#include <stddef.h>
#include <string.h>
#include <nos/multiboot.h>
#include <nos/vmm.h>
```

Data Structures

- struct page_table_t
- struct page_dir_t

Macros

- #define TABLES_PER_DIR 1024

  < Virtual memory management main macros
- #define PAGES_PER_TABLE 1024

  1 KB
- #define PAGE_SIZE 4096

  4 KB
- #define KERNEL_ADDR 0x50000
- #define PD_INDEX(addr) ((addr) >> 22)
- #define PT_INDEX(addr) (((addr) >> 12) & 0x3FF) /* max index 1023 (0x3FF) */
- #define PAGE_PADDRESS(dir_entry) ((*dir_entry) & ~0xFFF)
- #define SET_ATTRIBUTE(entry, attr) (*entry |= attr)
- #define CLEAR_ATTRIBUTE(entry, attr) (*entry &= ~attr)
- #define TEST_ATTRIBUTE(entry, attr) (*entry & attr)
- #define SET_FRAME(entry, addr) (*entry = (*entry & ~ 0x7FFFF000) | addr)

  Flags for Page Table Entry (PTE).

Enumerations

- enum PAGE_TABLE_FLAGS {
  PTE_PRESENT = 0x01 , PTE_READ_WRITE = 0x02 , PTE_USER = 0x04 , PTE_WRITE_THROUGH = 0x08 ,
  PTE_CACHE_DISABLE = 0x10 , PTE_ACCESSED = 0x20 , PTE_DIRTY = 0x40 , PTE_PAT = 0x80 ,
  PTE_GLOBAL = 0x100 , PTE_FRAME = 0x7FFFF000 }
- enum PAGE_DIR_FLAGS {
  PDE_PRESENT = 0x01 , PDE_READ_WRITE = 0x02 , PDE_USER = 0x04 , PDE_WRITE_THROUGH = 0x08 ,
  PDE_CACHE_DISABLE = 0x10 , PDE_ACCESSED = 0x20 , PDE_DIRTY = 0x40 , PDE_PAGE_SIZE = 0x80 ,
  PDE_GLOBAL = 0x100 , PDE_PAT = 0x2000 , PDE_FRAME = 0x7FFFF000 }

Functions

- u32 * vmm_get_pt_entry (page_table_t *pt, const u32 addr)
  Get the page table entry for a virtual address.
- u32 * vmm_get_pd_entry (page_dir_t *pd, const u32 addr)
  Get the page directory entry for a given virtual address.
- u32 * vmm_get_page (const u32 vaddr)
  Get the page entry for a given virtual address.
- void * vmm_page_alloc (u32 *page)
  Allocate a page.
- void vmm_free_page (u32 *page)
  Free a page.
- bool vmm_set_page_dir (page_dir_t *pd)
  Set the page directory.
- void vmm_flush_tlb_entry (u32 vaddr)
  Flush a TLB (Translation Lookaside Buffer) entry.
- bool vmm_map_page (void *paddr, void *vaddr)
  Map a physical address to a virtual address.
- void vmm_unmap_page (void *vaddr)
  Unmap a virtual address.
- bool vmm_init (void)
  Initialize the virtual memory manager.

4.47.1 Detailed Description

Contains declarations for virtual memory management.

This header file includes functions related to the virtual memory management.

Author

Alexander Kuzin (alkuzin)

Date

17.05.2024

4.47.2 Macro Definition Documentation

4.47.2.1 CLEAR_ATTRIBUTE

```
#define CLEAR_ATTRIBUTE(
              entry,
              attr ) (*entry &= ~attr)
```

4.47.2.2 KERNEL_ADDR

```
#define KERNEL_ADDR 0x50000
```

4.47.2.3 PAGE_PADDRESS

```
#define PAGE_PADDRESS(
              dir_entry ) ((*dir_entry) & ~0xFFF)
```

4.47.2.4 PAGE_SIZE

```
#define PAGE_SIZE 4096
```

4 KB

4.47.2.5 PAGES_PER_TABLE

```
#define PAGES_PER_TABLE 1024
```

1 KB

4.47.2.6 PD_INDEX

```
#define PD_INDEX(
              addr ) ((addr) >> 22)
```

4.47.2.7 PT_INDEX

```
#define PT_INDEX(
              addr ) (((addr) >> 12) & 0x3FF) /* max index 1023 (0x3FF) */
```

4.47.2.8 SET_ATTRIBUTE

```
#define SET_ATTRIBUTE(
              entry,
              attr ) (*entry |= attr)
```

4.47.2.9 SET_FRAME

```
#define SET_FRAME(
              entry,
              addr ) (*entry = (*entry & ~ 0x7FFFF000) | addr)
```

Flags for Page Table Entry (PTE).

4.47.2.10 TABLES_PER_DIR

```
#define TABLES_PER_DIR 1024
```

< Virtual memory management main macros

1 KB

4.47.2.11 TEST_ATTRIBUTE

```
#define TEST_ATTRIBUTE(
              entry,
              attr ) (*entry & attr)
```

4.47.3 Enumeration Type Documentation

4.47.3.1 PAGE_DIR_FLAGS

```
enum PAGE_DIR_FLAGS
```

Enumerator

PDE_PRESENT	
PDE_READ_WRITE	
PDE_USER	
PDE_WRITE_THROUGH	
PDE_CACHE_DISABLE	
PDE_ACCESSED	
PDE_DIRTY	
PDE_PAGE_SIZE	
PDE_GLOBAL	
PDE_PAT	
PDE_FRAME	

4.47.3.2 PAGE_TABLE_FLAGS

```
enum PAGE_TABLE_FLAGS
```

Enumerator

PTE_PRESENT	
PTE_READ_WRITE	
PTE_USER	
PTE_WRITE_THROUGH	
PTE_CACHE_DISABLE	
PTE_ACCESSED	
PTE_DIRTY	
PTE_PAT	
PTE_GLOBAL	
PTE_FRAME	

4.47.4 Function Documentation

4.47.4.1 vmm_flush_tlb_entry()

```
void vmm_flush_tlb_entry (
            u32 vaddr )
```

Flush a TLB (Translation Lookaside Buffer) entry.

The Translation Lookaside Buffer (TLB) is a cache memory in a computer system that stores recent translations of virtual memory to physical memory addresses.

Parameters

in	*vaddr*	- given virtual address to flush.

4.47.4.2 vmm_free_page()

```
void vmm_free_page (
            u32 * page )
```

Free a page.

Parameters

in	*page*	- given pointer to the page to be freed.

4.47.4.3 vmm_get_page()

```
u32 * vmm_get_page (
            const u32 vaddr )
```

Get the page entry for a given virtual address.

Parameters

in	*vaddr*	- given virtual address.

Returns

pointer to the page entry.

4.47.4.4 vmm_get_pd_entry()

```
u32 * vmm_get_pd_entry (
            page_dir_t * pd,
            const u32 addr )
```

Get the page directory entry for a given virtual address.

Parameters

in	*pd*	- given pointer to page directory.
in	*addr*	- given address.

Returns

pointer to the page directory entry in case of success.

null pointer otherwise.

4.47.4.5 vmm_get_pt_entry()

```
u32 * vmm_get_pt_entry (
            page_table_t * pt,
            const u32 addr )
```

Get the page table entry for a virtual address.

Parameters

in	*pt*	- given pointer to page table.
in	*addr*	- given address.

Returns

pointer to the page table entry in case of success.

null pointer otherwise.

4.47.4.6 vmm_init()

```
bool vmm_init (
            void  )
```

Initialize the virtual memory manager.

Returns

true - if initialization is successfull.

false - otherwise.

4.47.4.7 vmm_map_page()

```
bool vmm_map_page (
            void * paddr,
            void * vaddr )
```

Map a physical address to a virtual address.

Parameters

in	*paddr*	- given physical address.
out	*vaddr*	- given virtual address.

Returns

true - if translation is successfull.

false - otherwise.

4.47.4.8 vmm_page_alloc()

```
void * vmm_page_alloc (
            u32 * page )
```

Allocate a page.

Parameters

in	*page*	- given page pointer.

Returns

pointer to allocated page.

4.47.4.9 vmm_set_page_dir()

```
bool vmm_set_page_dir (
            page_dir_t * pd )
```

Set the page directory.

Parameters

in	*pd*	- given pointer to the page directory.

Returns

true - if page directory is set successfully.

false - otherwise.

4.47.4.10 vmm_unmap_page()

```
void vmm_unmap_page (
            void * vaddr )
```

Unmap a virtual address.

Parameters

in	*vaddr*	- given virtual address to unmap.

4.48 vmm.h

Go to the documentation of this file.

```
00001 /* MIT License
00002  *
00003  * Copyright (c) 2024 Alexander (@alkuzin)
00004  *
00005  * Permission is hereby granted, free of charge, to any person obtaining a copy
00006  * of this software and associated documentation files (the "Software"), to deal
00007  * in the Software without restriction, including without limitation the rights
00008  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00009  * copies of the Software, and to permit persons to whom the Software is
00010  * furnished to do so, subject to the following conditions:
00011  *
00012  * The above copyright notice and this permission notice shall be included in all
00013  * copies or substantial portions of the Software.
00014  *
00015  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00016  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00017  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00018  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00019  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00020  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00021  * SOFTWARE. */
00022
00034 #ifndef _NOS_KERNEL_VMM_H_
00035 #define _NOS_KERNEL_VMM_H_
00036
00037 #include <stdint.h>
00038 #include <stddef.h>
00039 #include <string.h>
00040
00041 #include <nos/multiboot.h>
00042 #include <nos/vmm.h>
00043
00045 #define TABLES_PER_DIR  1024
00046 #define PAGES_PER_TABLE 1024
00047 #define PAGE_SIZE       4096
00048
00049 #define KERNEL_ADDR 0x50000 // TODO: move to other header file
00050
00051 #define PD_INDEX(addr) ((addr) >> 22)
00052 #define PT_INDEX(addr) (((addr) >> 12) & 0x3FF) /* max index 1023 (0x3FF) */
```

```
00053 #define PAGE_PADDRESS(dir_entry) ((*dir_entry) & ~0xFFF)
00054 #define SET_ATTRIBUTE(entry, attr) (*entry |= attr)
00055 #define CLEAR_ATTRIBUTE(entry, attr) (*entry &= ~attr)
00056 #define TEST_ATTRIBUTE(entry, attr) (*entry & attr)
00057 #define SET_FRAME(entry, addr) (*entry = (*entry & ~ 0x7FFFF000) | addr)
00058
00060 typedef enum {
00061     PTE_PRESENT       = 0x01,
00062     PTE_READ_WRITE    = 0x02,
00063     PTE_USER          = 0x04,
00064     PTE_WRITE_THROUGH = 0x08,
00065     PTE_CACHE_DISABLE = 0x10,
00066     PTE_ACCESSED      = 0x20,
00067     PTE_DIRTY         = 0x40,
00068     PTE_PAT           = 0x80, /* PAT - Page Attribute Table */
00069     PTE_GLOBAL        = 0x100,
00070     PTE_FRAME         = 0x7FFFF000
00071 }PAGE_TABLE_FLAGS;
00072
00074 typedef enum {
00075     PDE_PRESENT       = 0x01,
00076     PDE_READ_WRITE    = 0x02,
00077     PDE_USER          = 0x04,
00078     PDE_WRITE_THROUGH = 0x08,
00079     PDE_CACHE_DISABLE = 0x10,
00080     PDE_ACCESSED      = 0x20,
00081     PDE_DIRTY         = 0x40,
00082     PDE_PAGE_SIZE     = 0x80, /* 0 - 4 KB page, 1 - 4 MB page */
00083     PDE_GLOBAL        = 0x100,
00084     PDE_PAT           = 0x2000,
00085     PDE_FRAME         = 0x7FFFF000
00086 }PAGE_DIR_FLAGS;
00087
00089 typedef struct {
00090     u32 entries[PAGES_PER_TABLE];
00091 } page_table_t;
00092
00094 typedef struct {
00095     u32 entries[TABLES_PER_DIR];
00096 } page_dir_t;
00097
00106 u32 *vmm_get_pt_entry(page_table_t *pt, const u32 addr);
00107
00116 u32 *vmm_get_pd_entry(page_dir_t *pd, const u32 addr);
00117
00124 u32 *vmm_get_page(const u32 vaddr);
00125
00132 void *vmm_page_alloc(u32 *page);
00133
00139 void vmm_free_page(u32 *page);
00140
00148 bool vmm_set_page_dir(page_dir_t *pd);
00149
00159 void vmm_flush_tlb_entry(u32 vaddr);
00160
00169 bool vmm_map_page(void *paddr, void *vaddr);
00170
00176 void vmm_unmap_page(void *vaddr);
00177
00184 bool vmm_init(void);
00185
00186 #endif /* _NOS_KERNEL_VMM_H_ */
```

Index