Writing A Page Frame Allocator

From OSDev Wiki

This page is a work in progress and may thus be incomplete. Its content may be changed in the near future.

Difficulty level

▮▮▯▯
Medium

Introduction

This tutorial will attempt to show you how to write a simple page frame allocator for the x86 CPU. The language used is C and we are using standard paging with 4 KiB pages. The page frame allocator will allocate frames with the first frame starting right after the end of the kernel.

The Method

Each frame shall be managed with a byte map (for the sake of simplicity): a value of 0x01 for used pages and a value of 0x00 for unused pages. To allocate a page, all that is needed is to search through the array for a free page and then mark it as used. You may have noticed that this would be very inefficient, having to search through a possible number of 1048319 pages. To speed up the allocation process the allocator will allocate 20 page frames at a time, so most of the time all that will have been done before hand. Alocatting a new page is simply a matter of getting a page frame off an array of preallocated frames.

The Implementation

First of all, we will need something in the linker script to tell us where the end of our kernel is.

```
ENTRY (loader)

SECTIONS{
    . = 0x00100000;

    .text :{
        text_start = .;
        *(.text)
    }

    .rodata ALIGN (0x1000) : {
        *(.rodata)
    }

.data ALIGN (0x1000) : {

   *(.data)
   end_data = .;
}


    .bss : {
        sbss = .;
        *(COMMON)
        *(.bss)
        ebss = .;
        endkernel = .;
    }


}
```

The variable endkernel will be declared in the kernel as:

```
extern uint32_t endkernel;
```

The variable itself has no value, it is the address of the variable that we are using. endkernel will be used to calculate the address of the first page frame after the kernel. The code for searching through the array is also very simple:

```
static pageframe_t kalloc_frame_int()
{
        uint32_t i = 0;
        while(frame_map[i] != FREE)
        {
                i++;
                if(i == npages)
                {
                        return(ERROR);
                }
        }
        frame_map[i] = USED;
        return(startframe + (i*0x1000));//return the address of the page
        //in the array
}
```

The last function used calls kalloc_frame_int every 20 page frame allocations:

```
pageframe_t kalloc_frame()
{
        static uint8_t allocate = 1;//whether or not we are going to allc
        static uint8_t pframe = 0;
        pageframe_t ret;

        if(pframe == 20)
        {
                allocate = 1;
        }
        if(allocate == 1)
        {
                for(int i = 0; i<20; i++)
                {
                        pre_frames[i] = kalloc_frame_int();
                }
                pframe = 0;
                allocate = 0;
        }
        ret = pre_frames[pframe];
        pframe++;
        return(ret);
}
```

Freeing the page frame is simply a matter of reversing the process used to get the page frame.

```
void kfree_frame(pageframe_t a)
{
                a = a - startframe;//get the offset from the first frame
                if(a == 0)//in case it is the first frame we are freeing
                {
                       uint32_t index = (uint32_t)a;
                       frame_map[index] = FREE;
                }
                else{
                       a = a;//divide by 4kb to get the index to declare
                       uint32_t index = ((uint32_t)a)/0x1000;
                       frame_map[index] = FREE;
                    }
}
```

Retrieved from "http://wiki.osdev.org/index.php?title=Writing_A_Page_Frame_Allocator&oldid=17167"
Categories: Level 2 Tutorials | In Progress | Tutorials

- This page was last modified on 1 December 2014, at 08:50.
- This page has been accessed 11,798 times.