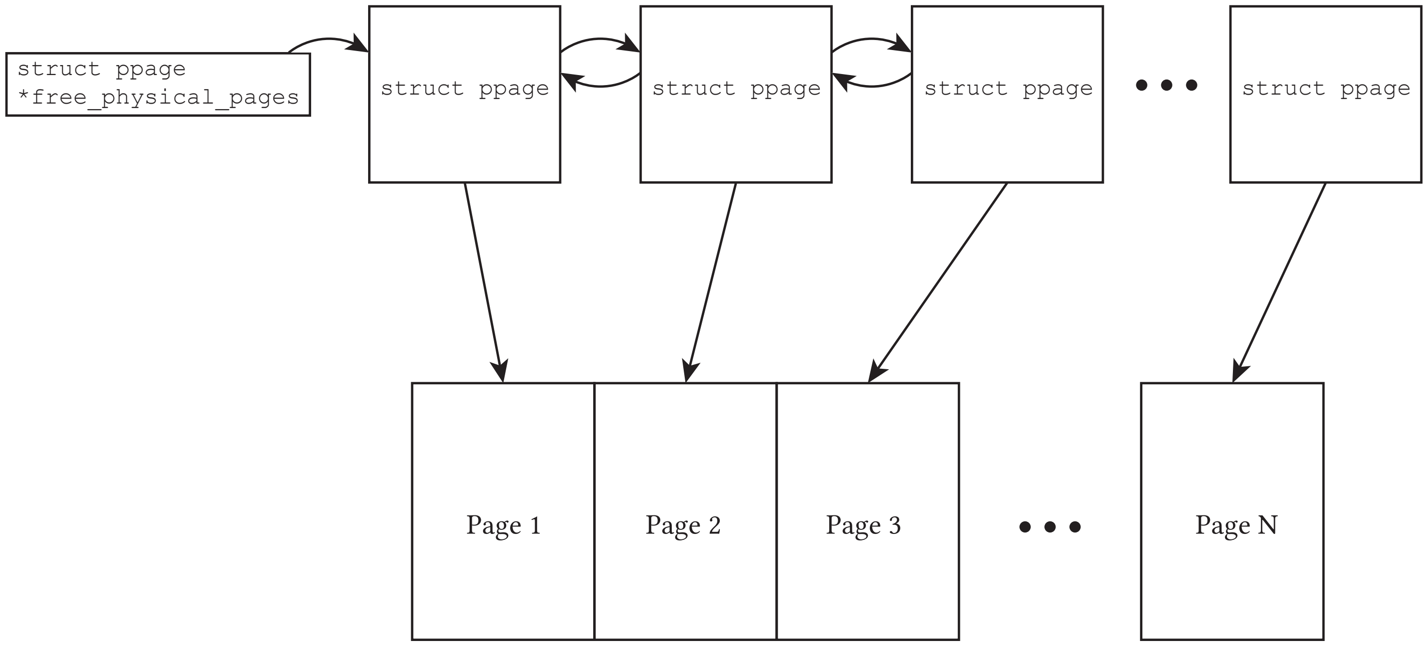
CS 310 Homework 5: Page Frame Allocator

Fall 2024

September 25, 2024

1 Introduction

As we discussed in class, the operating system assigns memory to processes in units called pages, which are blocks of a fixed length—usually 4kbytes on i386. In our operating system, we are going to divide physical memory up into 4 kbyte blocks, each of which can be assigned to a particular process. We will track each physical page in memory using a data structure that points to its physical address. Those data structures will initially be linked together in a linked list called `free_physical_pages` which will track the physical pages that are not allocated to any process.



When we need to allocate a physical page for a process, we will unlink the data structure for one physical page from the free list.

Deliverables

1. Create new files `page.c` and `page.h` in your kernel. Make sure that `page.c` is being compiled by your Makefile..
2. Inside `page.h` define `struct ppage` as below:

```
struct ppage {
  struct ppage *next;
  struct ppage *prev;
  void *physical_addr;
};
```

3. Inside `page.c`, statically allocate an array of `struct ppage`s:

   ```
   struct ppage physical_page_array[128]; // 128 pages, each 2mb in length covers 256 megs of memory
   ```

4. Inside `page.c` create a function that initializes the list of of free physical page structures:

   ```
   void init_pfa_list(void) {

   }
   ```

5. Inside `page.c`, create the following functions to allocate and free physical pages:

   ```
   struct ppage *allocate_physical_pages(unsigned int npages) {

   }

   void free_physical_pages(struct ppage *ppage_list){

   }
   ```

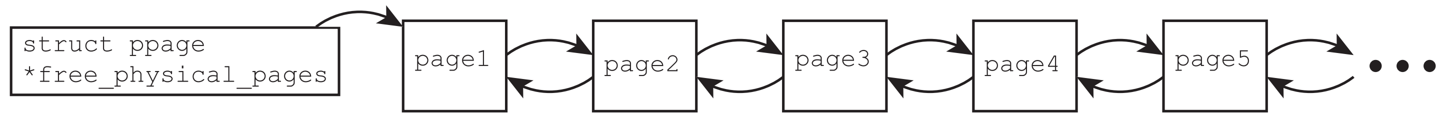
2 Implementation Details

All of the functions that you write for your page frame allocator are going to call your linked list functions to actually link and unlink elements from the list.

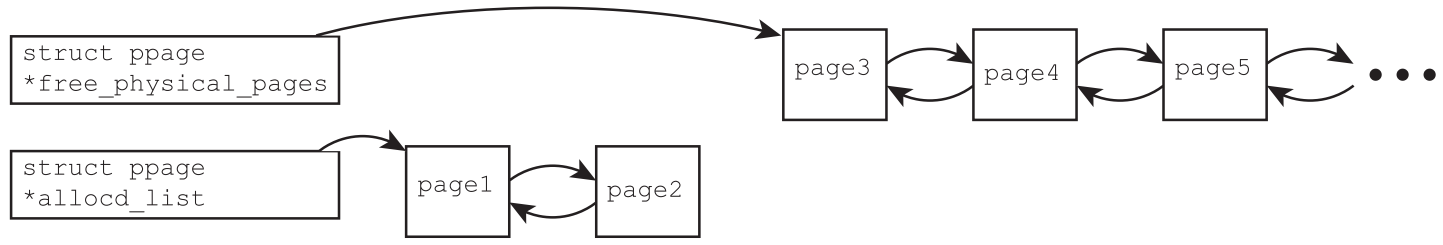
`init_pfa_list` will initialize the linked list of free pages. It will loop through every element of your statically allocated `physical_page_array` and link it into the list. In addition to linking each page structure into the free list, it will need to initialize the `physical_addr` member of the struct with the address of the page that it points to.

`allocate_physical_pages` allocates one or more physical pages from the list of free pages. The input to this function `npages` specifies the number of pages to allocate. The function will unlink the requested number of pages from the free list and create a new separate list. It will return a pointer to the new list (called `allocd_list` in the diagram below).

Before calling `allocate_physical_pages`:



After calling `allocate_physical_pages` with `npages = 2`:



`free_physical_pages` returns a list of physical pages to the free list. Basically does the opposite of `allocate_physical_pages`.

3 Further Reading

For more info, check the following resources:

- `https://wiki.osdev.org/Page_Frame_Allocation`
- `https://wiki.osdev.org/Writing_A_Page_Frame_Allocator`