Sistemi Operativi[1]

Mattia Monga

Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2015/16

[1]

Lezione XXI: Gestione della memoria in JOS

Indirizzi

Nei manuali x86 si parla di 3 tipologie di indirizzi

virtuali quando sono relativi ad un segmento: un puntatore C è un *offset*

lineare selettore di segmento + offset permette di calcolare un indirizzo nello spazio di indirizzamento (virtuale) lineare 0–4GB

fisico l'indirizzo lineare è "mappato" su un indirizzo fisico dalla MMU (che non può essere saltata!)

Come manipolare gli indirizzi

Segmentazione e MMU non possono essere saltati: il programmatore "vede" esclusivamente indirizzi virtuali.

- JOS configura tutti i segmenti (in `boot/boot.S` tramite la prima GDT) in `0-0xffffffff` (0–4GB), quindi il segmento può essere ignorato
- Quando serve manipolare indirizzi fisici (che non possono essere dereferenziati) devono essere usati *numeri* che sarà utile contrassegnare con il tipo physaddr_t
- Un numero che può essere dereferenziato (perché si tratta di un indirizzo virtuale) verrà contrassegnato con uintptr_t e per derefenziarlo come `T` va interpretato come `T*`.

Il mapping iniziale

I kernel sono generalmente caricati a un indirizzo (lineare) alto, p.es. `0xf0100000` (3,75GB), che potrebbe perfino non esistere nello spazio fisico.

- il programmatore del kernel (e il programma!) usa `0xf0100000` (virtuale)
- il boot loader carica il kernel all'indirizzo `0x00100000`
- il boot loader istruisce la MMU perché mappi `0xf0100000` $\rightarrow$ `0x00100000`

le prime page table

- La page table 'zeresima' in `boot/boot.S` configura il mapping *identità*, quindi indirizzi lineari uguali a fisici.
- La prima vera page table è in `kern/entrypgdir.c`

lineare	fisico
0xf0000000 (KERNBASE)	0x00000000
...	...
0xf0400000	0x00400000 (4MB)
0x00000000	0x00000000
...	...
0x00400000	0x00400000 (4MB)
*	eccezione

Macro che sostituiscono la MMU

0xf0000000 == KERNBASE $\rightarrow$ 0x00000000

0xf0100000 == KERNBASE + 1MB

0xf0400000 == KERNBASE + 4MB $\rightarrow$ 0x00400000

Alla fine del lab2 verranno mappati 256MB. Si noti che esiste una relazione semplice fra fisico e lineare: quando serve il programmatore può calcolare l'indirizzo lineare aggiungendo KERNBASE al fisico. Per farlo meglio usare KADDR (e PADDR per l'inverso) che controllano che il numero cui si applica sia sensato.

Le strutture dati per la gestione della memoria


```
1 struct PageInfo *pages;  // Physical page state array
2 static struct PageInfo *page_free_list; // Free list of physical pages
```

(Lo static garantisce che page_free_list sia "privata" del file `kern/pmap.c`. Analogamente la variabile nextfree è privata alla funzione boot_alloc, anche se la durata del suo valore è analoga a quella di una variabile globale: si mantiene fra una chiamata e l'altra)

1. L'array npages viene allocata inizialmente con boot_alloc
2. Viene inizializzata con page_init; una pagina è libera se fa parte della lista collegata page_free_list
3. L'allocazione poi deve avvenire sempre con page_alloc

Il reference count di una pagina (quante pagine virtuali vengono mappate su di essa) è aggiornato da page_insert. Per altri usi occorre farlo a mano.