

Trabajos Prácticos con JOS

Federico del Mazo - 100029

Rodrigo Souto - 97649

Trabajos Prácticos con JOS

Respuestas teóricas de los distintos trabajos prácticos/labs de Sistemas Operativos (75.08).

TP1: Memoria virtual en JOS (26/04/2019)

Memoria física: boot_alloc_pos

1. Incluir: Un cálculo manual de la primera dirección de memoria que devolverá boot_alloc() tras el arranque. Se puede calcular a partir del binario compilado (obj/kern/kernel), usando los comandos readelf y/o nm y operaciones matemáticas.

Truncando la salida de ambos comandos (con **grep**), vemos las siguientes líneas:

```
sisop_2019a_delmazo_souto TP1 % readelf -s obj/kern/kernel | grep end
112: f0117950      0 NOTYPE GLOBAL DEFAULT      6 end
sisop_2019a_delmazo_souto TP1 % nm obj/kern/kernel | grep end
f0117950 B end
```

Como podemos ver en ambos casos, la dirección de memoria que recibe boot_alloc() es f0117950 (en decimal, 4027677008). A este valor, la función (en su primera llamada) lo redondea a 4096 (PGSIZE) llamando a ROUNDUP(a,n). Además de devolver ese valor redondeado, guarda la variable nextfree en una página más de lo recibido.

Por ende, el valor devuelto será el de ROUNDUP(4027677008, 4096). Esta función, como indica su documentación, redondea a a al múltiplo más cercano de n. Este múltiplo será 4027678720, que está más cerca que el siguiente múltiplo (4027682816). Para confirmarlo desde la práctica, se traducen a Python los cálculos que utiliza la función de redondeo:

```
sisop_2019a_delmazo_souto TP1 % python3
Python 3.6.7 |Anaconda, Inc.| (default, Oct 23 2018, 19:16:44)
>>> a = 0xf0117950
>>> n = 4096
>>> def rounddown(a,n): return a - a % n
>>> def roundup(a,n): return rounddown(a + n - 1, n)
>>> res = roundup(int(a),n)
>>> res
4027678720
>>> hex(res)
'0xf0118000'
```

2. Incluir: Una sesión de GDB en la que, poniendo un breakpoint en la función boot_alloc(), se muestre el valor de end y nextfree al comienzo y al fin de esa primera llamada a boot_alloc().

```
sisop_2019a_delmazo_souto TP1 % make gdb
gdb -q -s obj/kern/kernel -ex 'target remote 127.0.0.1:26000' -n -x .gdbinit
Reading symbols from obj/kern/kernel...done.
Remote debugging using 127.0.0.1:26000
0x0000fff0 in ?? ()
```

```

(gdb) break boot_alloc
Breakpoint 1 at 0xf0100a58: file kern/pmap.c, line 89.
(gdb) continue
Continuing.
The target architecture is assumed to be i386
=> 0xf0100a58 <boot_alloc>: push    %ebp

Breakpoint 1, boot_alloc (n=4096) at kern/pmap.c:89
89 {
(gdb) print (char*) &end
$1 = 0xf0117950 ""
(gdb) watch &end
Watchpoint 2: &end
(gdb) watch nextfree
Hardware watchpoint 3: nextfree
(gdb) continue
Continuing.
=> 0xf0100aac <boot_alloc+84>: jmp     0xf0100a68 <boot_alloc+16>

Hardware watchpoint 3: nextfree

Old value = 0x0
New value = 0xf0118000 ""
0xf0100aac in boot_alloc (n=4096) at kern/pmap.c:100
100      nextfree = ROUNDUP((char *) end, PGSIZE);
(gdb) continue
Continuing.
=> 0xf0100a81 <boot_alloc+41>: mov     0xf0117944,%edx

Hardware watchpoint 3: nextfree

Old value = 0xf0118000 ""
New value = 0xf0119000 ""
boot_alloc (n=4096) at kern/pmap.c:111
111      if (nextfree >= (char *) (KERNBASE + npages * PGSIZE)) {
(gdb) continue
Continuing.

```

Como se puede ver, se cumple todo lo planteado. `end` comienza en `0xf0117950`, luego `nextfree` se inicializa en el número ya redondeado `0xf0118000`, y finalmente se avanza una página, y queda `0xf0119000`.

Memoria física: `page_alloc`

1. Responder: ¿en qué se diferencia `page2pa()` de `page2kva()`?

Como bien indican sus nombres, `page2pa()` y `page2kva()` se diferencian en el valor de retorno. Ambas reciben una página física, pero `page2pa()` devuelve su dirección física (de tipo `physaddr_t`) mientras que `page2kva()` devuelve la dirección virtual (kernel virtual address), de tipo `void*`.

Incluso, `page2kva()` no es más que un llamado a `page2pa()` y luego a la función del preprocesador `KADDR()` que recibe una dirección física y devuelve la respectiva dirección virtual.

Large pages: `map_region_large`

1. Responder: ¿cuánta memoria se ahorró de este modo? ¿Es una cantidad fija, o depende de la memoria física de la computadora?