Dr. Kerneldev
or:
how I learned to stop worrying and love the pagefault

Γιάννης Τσιομπίκας

nuclear@member.fsf.org

4 Νοεμβρίου 2011

Previously on dr.kerneldev...

- Booting με το GRUB.
- VGA text mode driver & printf.
- Ισοπεδόνωντας το x86 segmented memory model.
- Virtual $\rightarrow$ physical page translation συνοπτικά.
- 8259 PIC (Programmable Interrupt Controller).
- Χειρισμός interrupts & exceptions.

Today's topics

- Memory management
- Timekeeping
- Task switching
- Processes

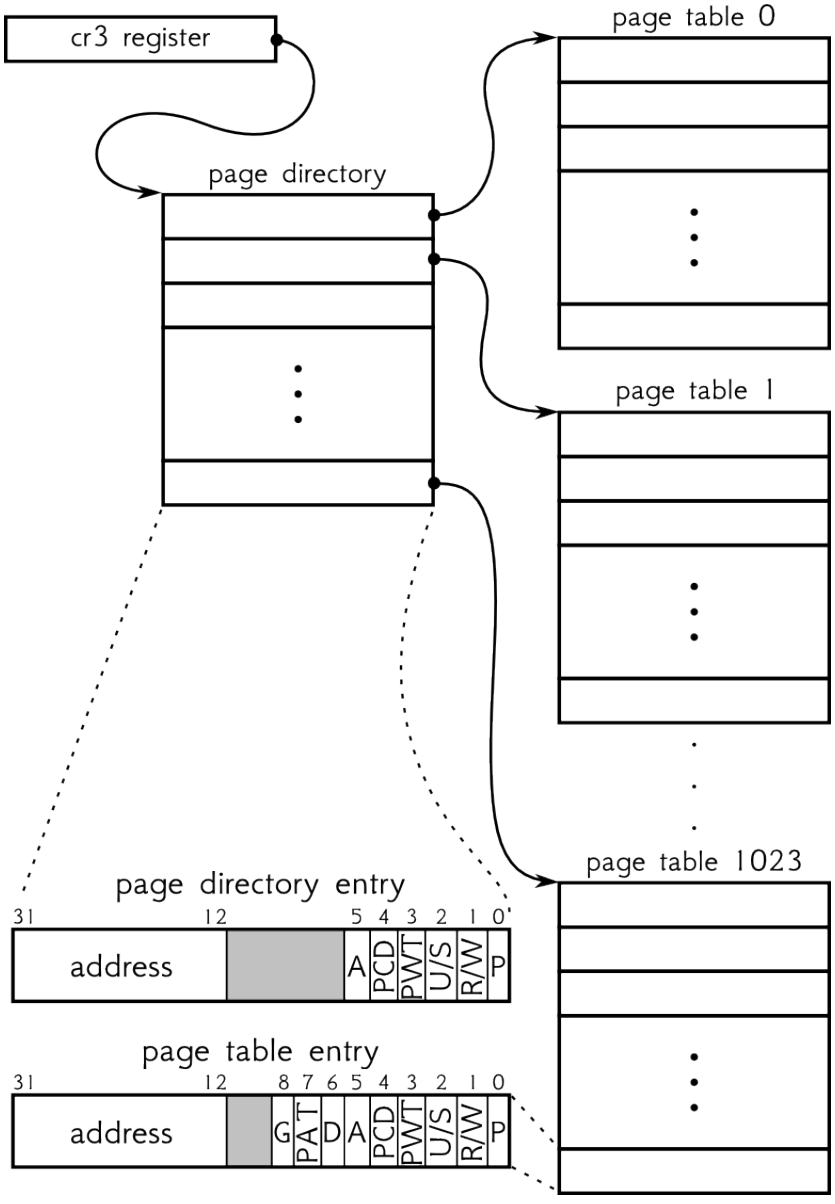
Memory Management

Physical Memory Management

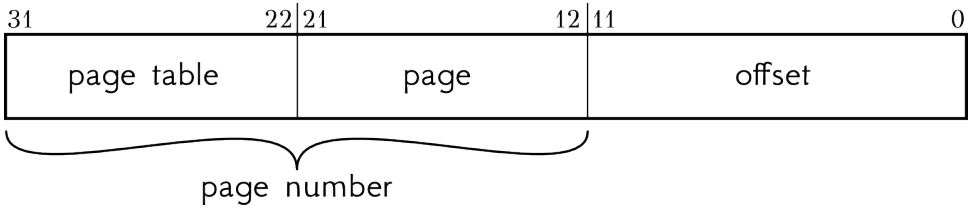
- Multiboot memory map (απο το GRUB)
- Bitmap allocator (alloc_phys_page / free_phys_page)

```
memory map:
  free: 0 - 9fc00 (654336 bytes)
  hole: 9fc00 - a0000 (1024 bytes)
  hole: e0000 - 100000 (131072 bytes)
  free: 100000 - bf780000 (3211264000 bytes)
  hole: bf780000 - bf78e000 (57344 bytes)
  hole: bf78e000 - bf7d0000 (270336 bytes)
  hole: bf7d0000 - bf7e0000 (65536 bytes)
  hole: bf7ed000 - bf800000 (77824 bytes)
  hole: bf800000 - c0000000 (8388608 bytes)
  hole: fee00000 - fee01000 (4096 bytes)
  hole: ffa00000 - ffffffff (6291455 bytes)
marking pages up to 120e7f (page: 288) inclusive as used
```

Page Translation



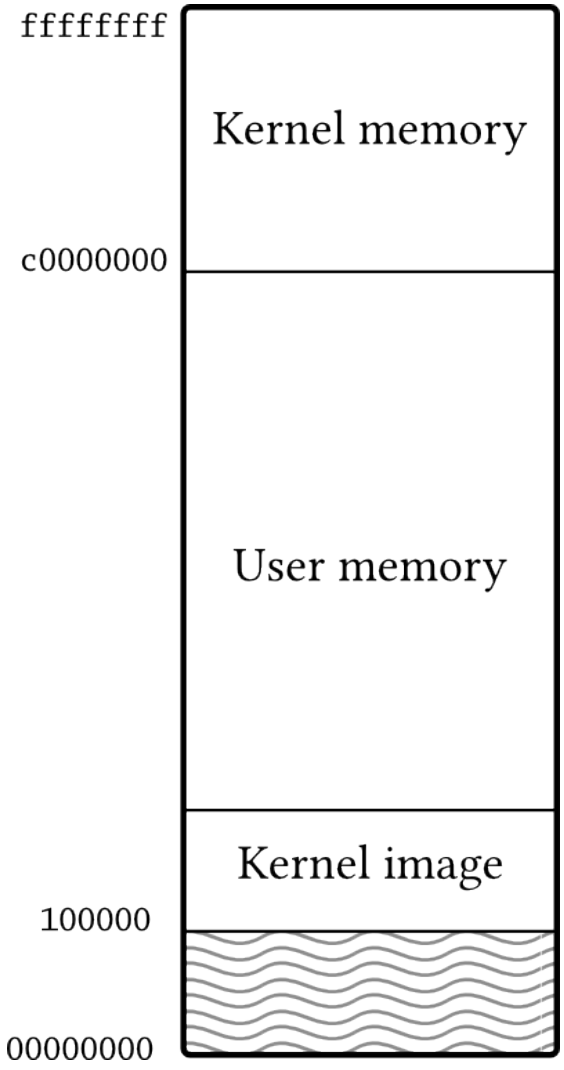
Virtual address translation



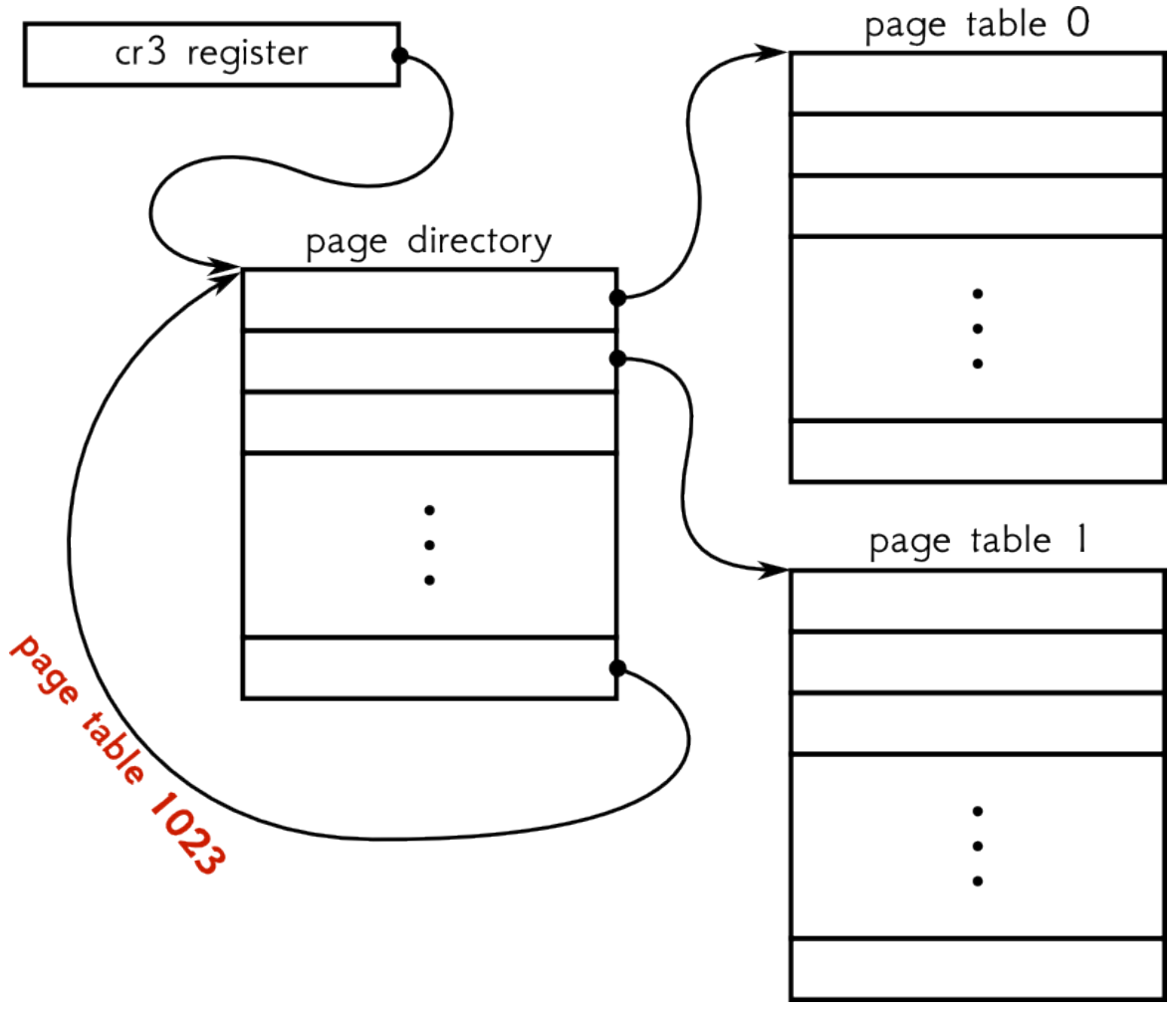
- Τα 10 ανώτερα bits $[22, 31]$ είναι index στο page directory (διαλέγουν page table).
- Τα επόμενα 10 bits $[12, 21]$ είναι index στο επιλεγμένο page table (διαλέγουν page).
- Τα κατώτερα 12 bits είναι το offset μέσα στο page.

Virtual Memory Management

- map_page / unmap_page
- TLB management
  (flush_tlb / flush_tlb_addr)
- Page range list allocator
  (pgalloc / pgfree)
- Higher level allocator
  (malloc / free)
- Page fault handling
  (more on that later)
- Recursive page tables
  (cont'd next slide...)



Recursive page tables



Timekeeping

8253 Programmable Interval Timer

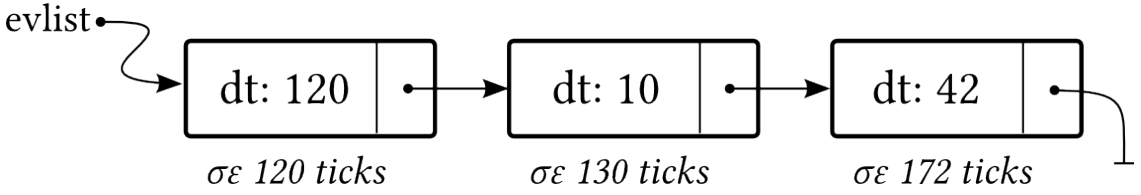
- Driven απο κρύσταλλο 1.193182 MHz
- 3 16bit binary counters
  - Channel 0: Γενικής χρήσεως, IRQ 0 στο underflow
  - Channel 1: DRAM refresh (unusable)
  - Channel 2: PC speaker
- Πολλαπλά modes λειτουργίας (one shot, rate, square wave, strobe...).

```
#define OSC_FREQ_HZ       1193182
#define TICK_FREQ_HZ      250

reload = DIV_ROUND(OSC_FREQ_HZ, TICK_FREQ_HZ);
```

Timer handling

- Global ticks counter (nticks).
- Λίστα με events, χρώνος σχετικός με προηγούμενο στη λίστα.

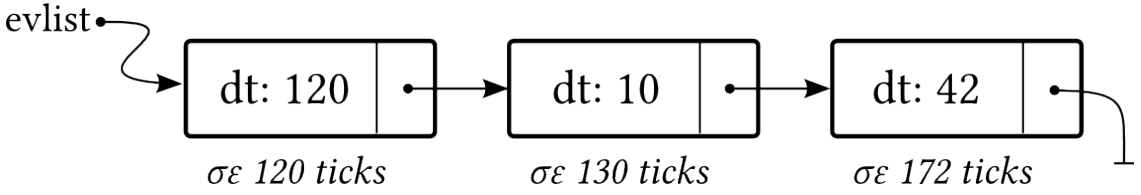


Interrupt handling

- increment `nticks`
- decrement `dt` του πρώτου event και wakeup αν είναι 0.
- call `schedule`

Timer handling

- Global ticks counter (nticks).
- Λίστα με events, χρώνος σχετικός με προηγούμενο στη λίστα.

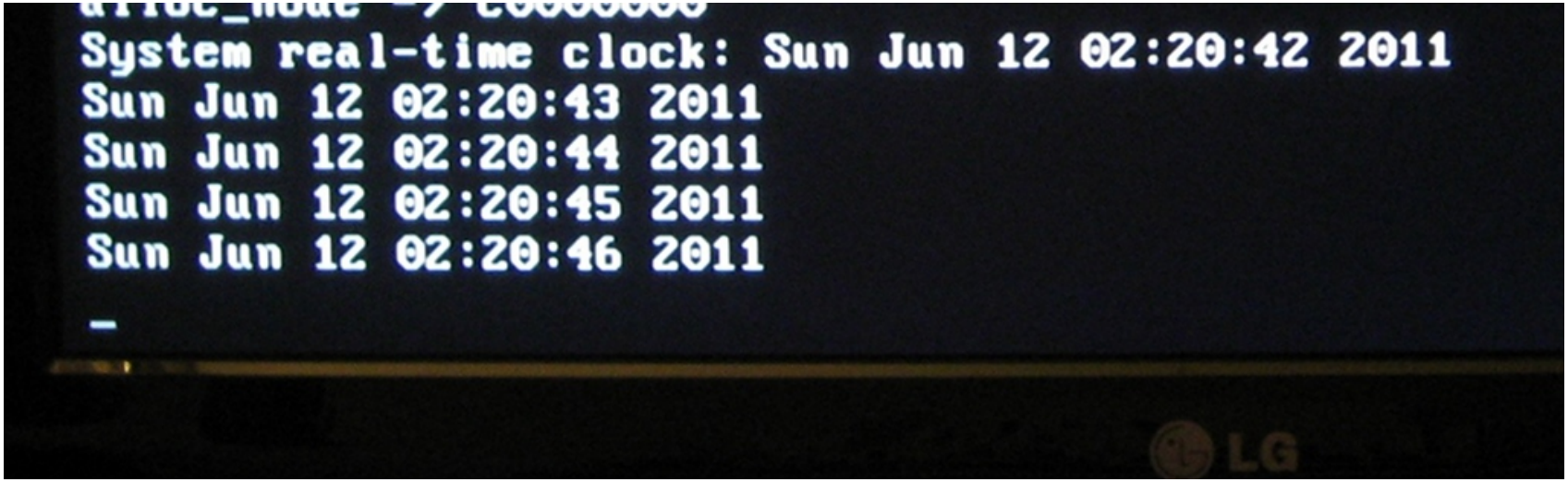


Interrupt handling

- increment `nticks`
- decrement `dt` του πρώτου event και wakeup αν είναι 0.
- call `schedule`

Real Time Clock

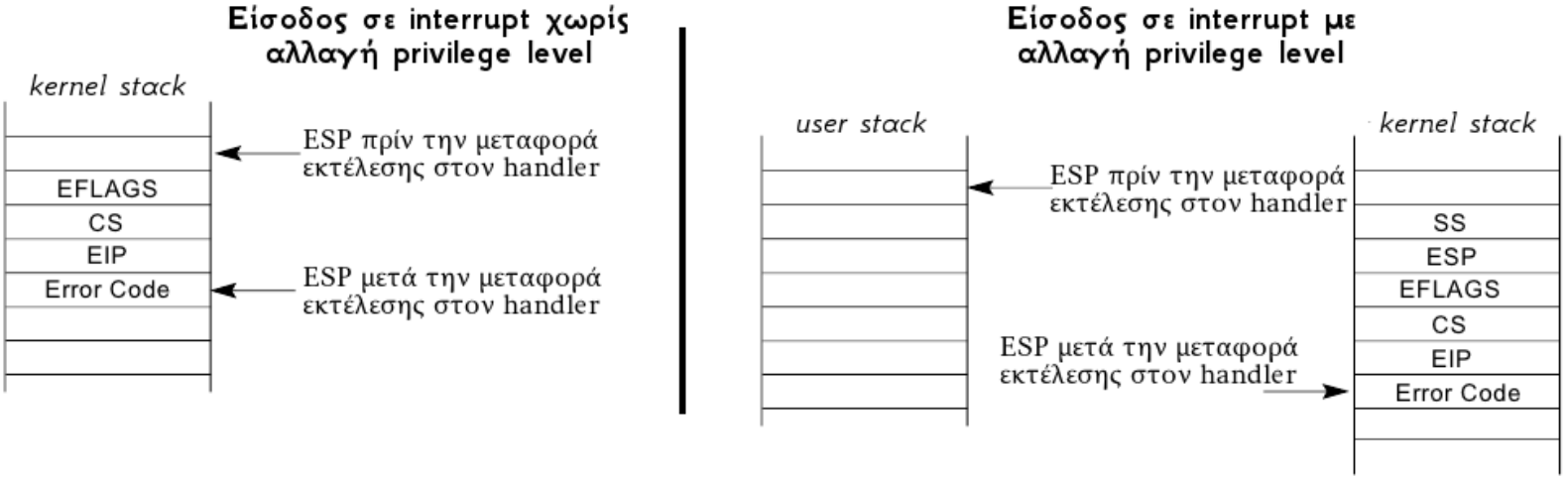
- RTC registers σε battery-backed static RAM (CMOS)
- Πιθανές μορφές δεδομένων στούς RTC registers, ορίζεται στον status register:
  - BCD ή decimal (συνήθως BCD !?@?!)
  - 24h ή 12h με high-order bit AM/PM
  - Έτος '2011' ή '11' (συνήθως '11' ?@?#!)
- Στον kernel: seconds από UNIX epoch & standard C mktime/asctime κ.τ.λ στην klibc.



Processes

Kernel entry / exit

- 4 priviledge levels $[0, 3]$
- Αλλαγή του priviledge level όταν φορτώνεται segment descriptor με διαφορετικό `dpl`.
  - interrupts (int/iret)
  - call gates (intersegment call/ret)
  - sysenter/sysexit and friends
- Κάθε priviledge level < 3 έχει δικό του stack (ss/esp) στο TSS.



System calls

- interrupt 128
- system call number $\rightarrow$ eax
- arguments $\rightarrow$ ebx, ecx, edx, esi, edi
- result $\rightarrow$ eax

```
sys_func[SYS_HELLO] = sys_hello;
sys_func[SYS_SLEEP] = sys_sleep;      /* timer.c */
sys_func[SYS_FORK] = sys_fork;        /* proc.c */
sys_func[SYS_EXIT] = sys_exit;        /* proc.c */
sys_func[SYS_WAITPID] = sys_waitpid; /* proc.c */
sys_func[SYS_GETPID] = sys_getpid;    /* proc.c */
sys_func[SYS_GETPPID] = sys_getppid; /* proc.c */
```

Scheduler

Συνάρτηση `schedule`:

- Διαλέγει πάντα το πρώτο process απο την λίστα.
- Άν εξαντλήσει το timeslice του, αφαιρείται απο μπροστά και μπαίνει πίσω.
- Καλεί την `context_switch`.
- Άν το runqueue είναι άδειο καλεί την `idle_proc` που κάνει halt την CPU.

Context Switching

Τα context switches ξεκινάν με κλήση στην `context_switch`, η οποία:

- Αλλάζει page tables (cr3).
- Θέτει το kernel stack του καινούριου user process στο TSS.
- Κάνει push το state του current process στο stack του.
- Καλεί την `switch_stack` η οποία *επιστρέφει στο καινούριο context*!
- Επαναφέρει το state απο το καινούριο stack.

Σημείωση

Μετά απο fork η `swich_stack` ΔΕΝ επιστρέφει στο ίδιο σημείο και δέν εκτελείται το υπόλοιπο της `context_switch`.

Context Switching

Τα context switches ξεκινάν με κλήση στην `context_switch`, η οποία:

- Αλλάζει page tables (cr3).
- Θέτει το kernel stack του καινούριου user process στο TSS.
- Κάνει push το state του current process στο stack του.
- Καλεί την `switch_stack` η οποία *επιστρέφει στο καινούριο context*!
- Επαναφέρει το state απο το καινούριο stack.

Σημείωση

Μετά απο fork η `swich_stack` ΔΕΝ επιστρέφει στο ίδιο σημείο και δέν εκτελείται το υπόλοιπο της `context_switch`.

Process creation - fork

Processes δημιουργούνται με το system call `fork`.

- Allocate entry στο process table.
- Allocate kernel stack για το process.
- Αντιγράφει το current interrupt frame στο καινούριο stack, ώστε επιστρέφωντας σε user space να συνεχιστεί η εκτέλεση απο το ίδιο σημείο.
- Αλλάζει την τιμή του eax στο interrupt frame σε 0.
- Τοποθετεί την διεύθυνση της `just_forked` στο καινούριο stack για να επιστρέψει εκεί η `switch_stacks`.
- Καλεί την `clone_vm` για να πάρει το καινούριο process, αντίγραφο της μνήμης του current process.
- Προσθέτει το καινούριο process στο runqueue.

Process creation - clone_vm

Κάθε process έχει ενα `vmmap` που περιέχει ένα `vm_page` sturcture για κάθε mapped virtual page στο memory space του.
Το vmmap είναι ένα balanced binary search tree, για γρήγορη αναζήτηση με βάση το page number.

Η `clone_vm`:

- Αντιγράφει τα page tables (για το user memory space) του current process αφού καθαρήσει τα write bits.
- Αντιγράφει το kernel κομμάτι του page directory ώστε να χρησιμοποιηθούν τα ίδια page tables για την μνήμη του πυρήνα σε όλα τα processes.
- Δημιουργεί καινούριο `vmmap` με pointers στα ίδια `vm_page` αφού αυξήσει το reference count τους (`nref`).

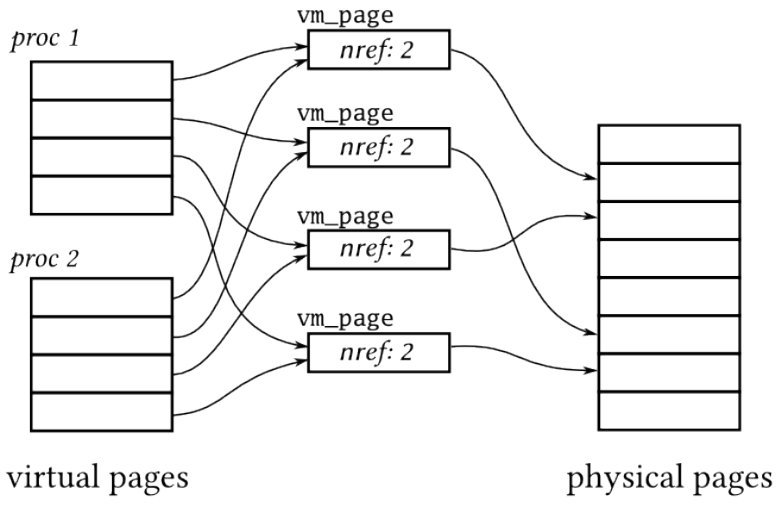
Copy on Write

Εγγραφή σε κάποιο απο τα κοινά pages σηκώνει pagefault. Ο page fault handler κάνει τα εξής:

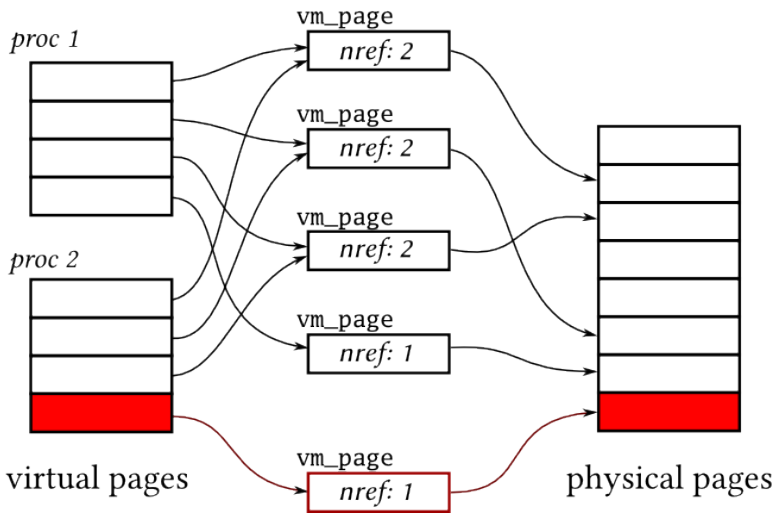
- Αναζητά το `vm_page` που αντιστοιχεί στο συγκεκριμένο page.
- Αν το `vm_page` είναι writable άλλα σηκώθηκε write fault πρόκειται για CoW fault.
- Στην οποία περίπτωση κάνει allocate καινούριο physical page και το κάνει map.
- Αντιγράφει τα περιεχόμενα του page που faultαρε στο καινούριο.
- Δημιουργεί καινούριο `vm_page` (με nref=0) και το αντικαθηστά το παλιό στο vmmap με το καινουριο.
- μειώνει το reference count του original `vm_page`.

Copy on Write - diagram

Μετά το fork όλα τα pages είναι κοινά...



Αντιγραφή μόνο σε απόπειρα εγγραφής.



First process

Το πρώτο process το κατασκευάζει η `start_first_proc`

- Γράφει το process image στη μνήμη.
- Κάνει allocate user stack και kernel stack.
- Τοποθετεί το kernel stack στο TSS.
- Του δίνει το υπάρχων page table.
- Δημειουργέι vmmap με βάση το υπάρχων page table.
- Το κάνει current και το προσθέτει στο runqueue.
- Κατασκευάζει ψέυτικο interrupt frame.
- καλεί την `intr_ret` για να κάνει fake επιστροφή απο interrupt στο καινούριο process, ώστε να γίνει η πρώτη αλλαγή priviledge level $0 \rightarrow 3$.

Blocking

- Τα processes blockάρουν σε kernel space καλώντας `wait` με την διεύθυνση αυτού που περιμένουν (wait channel).
- Η `wakeup` ξυπνάει όλα τα processes που περιμένουν σε ένα συγκεκριμένο wait channel.
- Hash table για αντιστοιχία wait channel $\rightarrow$ process.

Ερωτήσεις;

Links

- https://nuclear.mutantstargoat.com/hg/kern
- http://nuclear.sdf-eu.org/articles/kerneldev
- http://codelab.wordpress.com
- http://www.linuxinside.gr