



Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχανικών Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

4^η Εργαστηριακή Άσκηση:

Εικονική Μνήμη

Λειτουργικά Συστήματα Υπολογιστών
6ο Εξάμηνο, 2020-2021

Σύνοψη



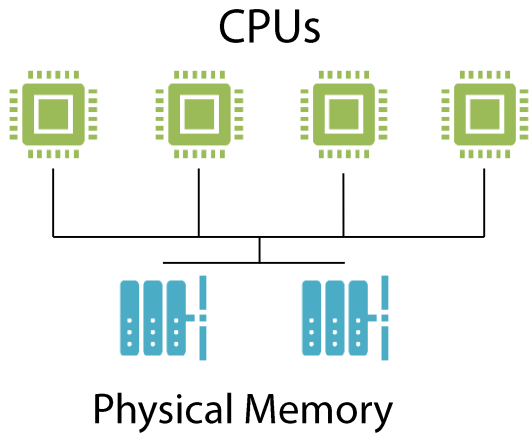
- ◆ Μηχανισμοί εικονικής μνήμης (virtual memory)
- ◆ Ζητούμενο 1: Κλήσεις συστήματος
 - ➔ Δημιουργία/διαχείριση απεικονίσεων (mmap() etc)
 - ➔ Εξέταση της απεικόνισης στη φυσική μνήμη
- ◆ Ζητούμενο 2: Parallel Mandelbrot με τη χρήση διεργασιών
 - ➔ Υλοποίηση σημαφόρων με διαμοιραζόμενη μνήμη
 - ➔ Υλοποίηση χωρίς συγχρονισμό με τη χρήση διαμοιραζόμενης μνήμης

Σύνοψη

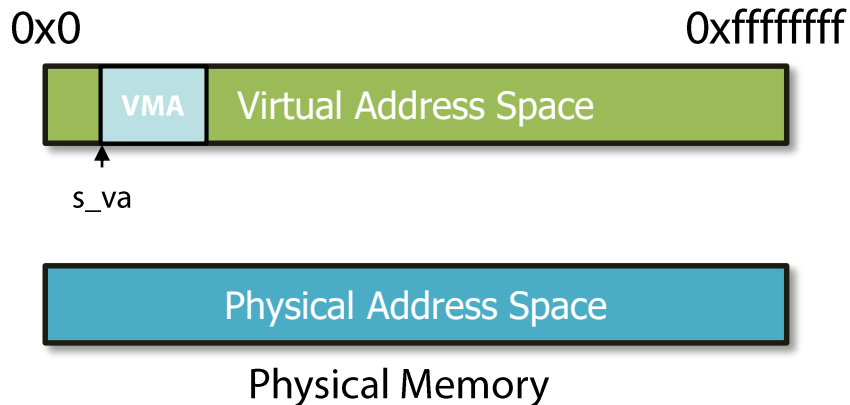


- ◆ Μηχανισμοί εικονικής μνήμης (virtual memory)
- ◆ Ζητούμενο 1: Κλήσεις συστήματος
 - ➔ Δημιουργία/διαχείριση απεικονίσεων (mmap() etc)
 - ➔ Εξέταση της απεικόνισης στη φυσική μνήμη
- ◆ Ζητούμενο 2: Parallel Mandelbrot με τη χρήση διεργασιών
 - ➔ Υλοποίηση σημαφόρων με διαμοιραζόμενη μνήμη
 - ➔ Υλοποίηση χωρίς συγχρονισμό με τη χρήση διαμοιραζόμενης μνήμης

Εικονική Μνήμη Επισκόπηση



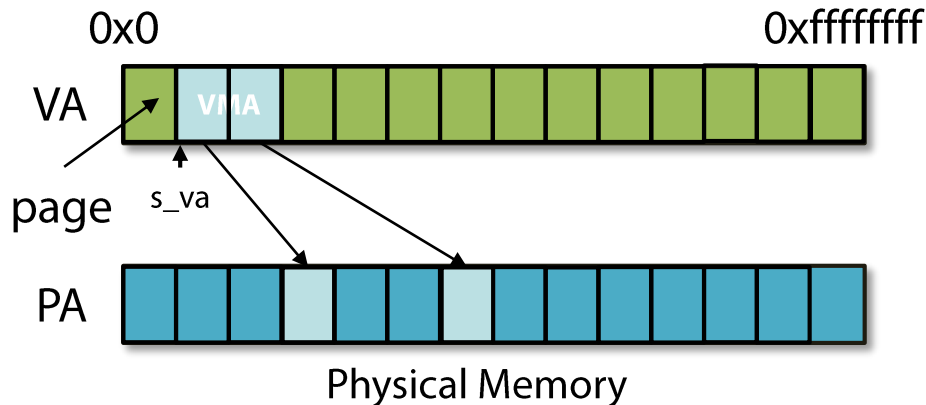
Εικονική Μνήμη Επισκόπηση



`s_va = malloc();` `libc`
`s_va = brk()/mmap();` `OS syscalls`

- ◆ Εικονική μνήμη: γραμμικός συνεχής χώρος διευθύνσεων
- ◆ Κλήσεις συστήματος → το ΛΣ δεσμεύει μια περιοχή εικονικών διευθύνσεων (virtual memory area – VMA)

Εικονική Μνήμη Επισκόπηση

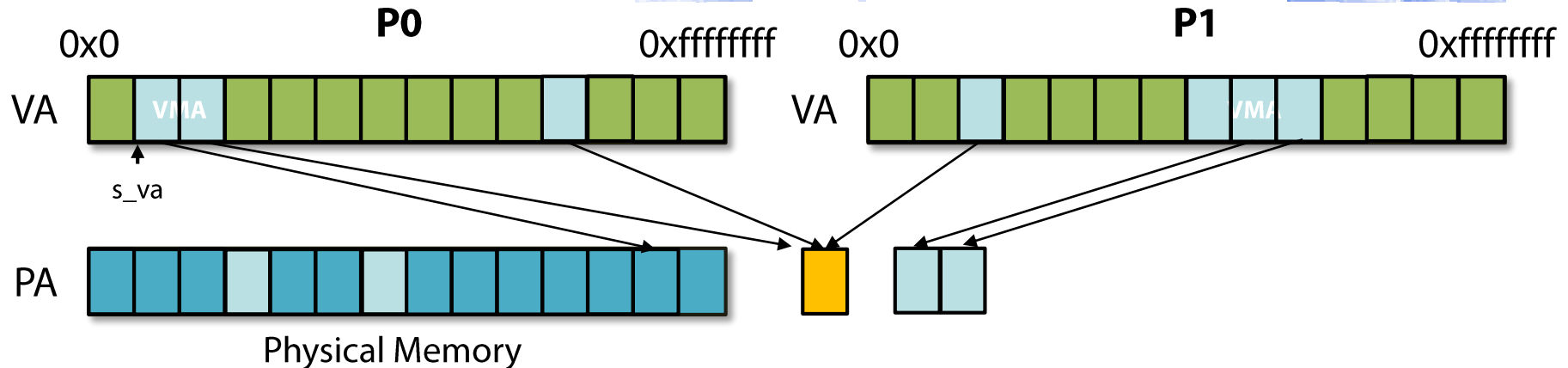


`s_va = malloc();` **libc**
`s_va = brk()/mmap();` **OS syscalls**

◆ Σελιδοποίηση

- ➔ Το ΛΣ διαχειρίζεται τη φυσική μνήμη και τις απεικονίσεις (VA-to-PA mappings) σε επίπεδο σελίδων (4K/2M/1G)

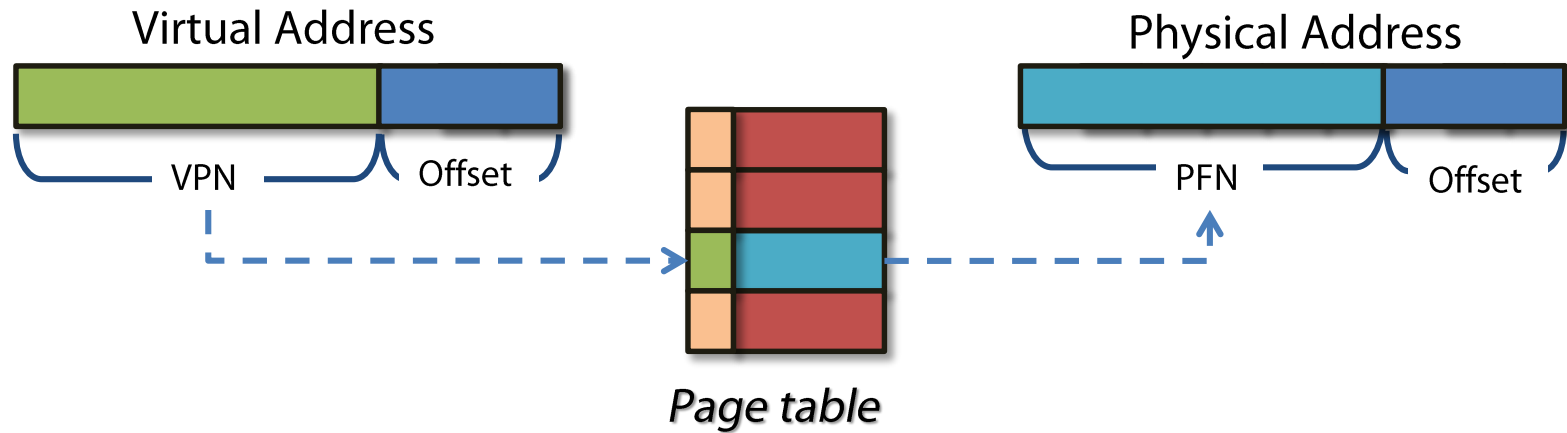
Εικονική Μνήμη Επισκόπηση



◆ Σελιδοποίηση

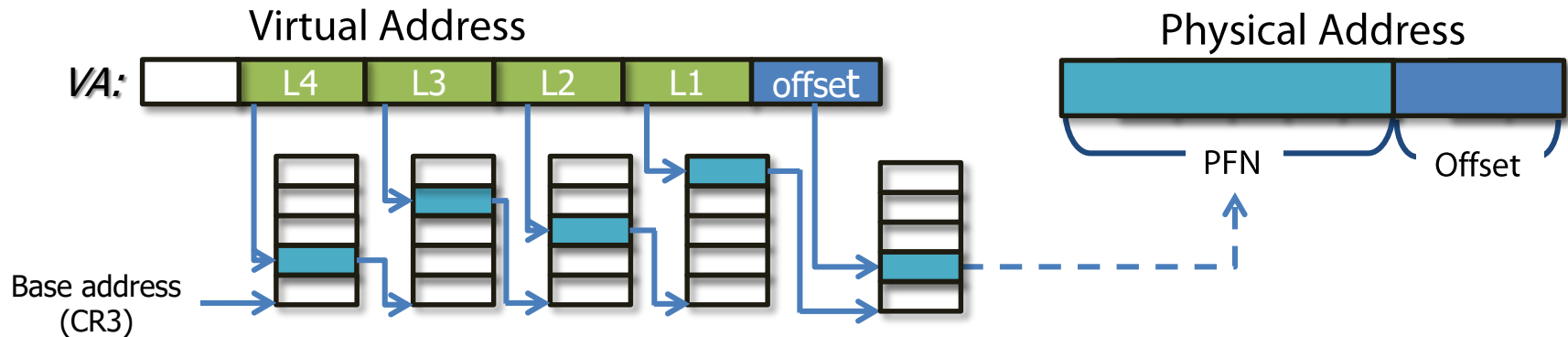
- ➔ Το ΛΣ διαχειρίζεται τη φυσική μνήμη και τις απεικονίσεις (VA-to-PA mappings) σε επίπεδο σελίδων (4K/2M/1G)
- ➔ Κάθε διεργασία έχει δικό της διακριτό γραμμικό χώρο εικονικών διευθύνσεων
- ➔ Μοιραζόμενη μνήμη, με αντιστοίχιση σε κοινή φυσική σελίδα

Εικονική Μνήμη Επισκόπηση



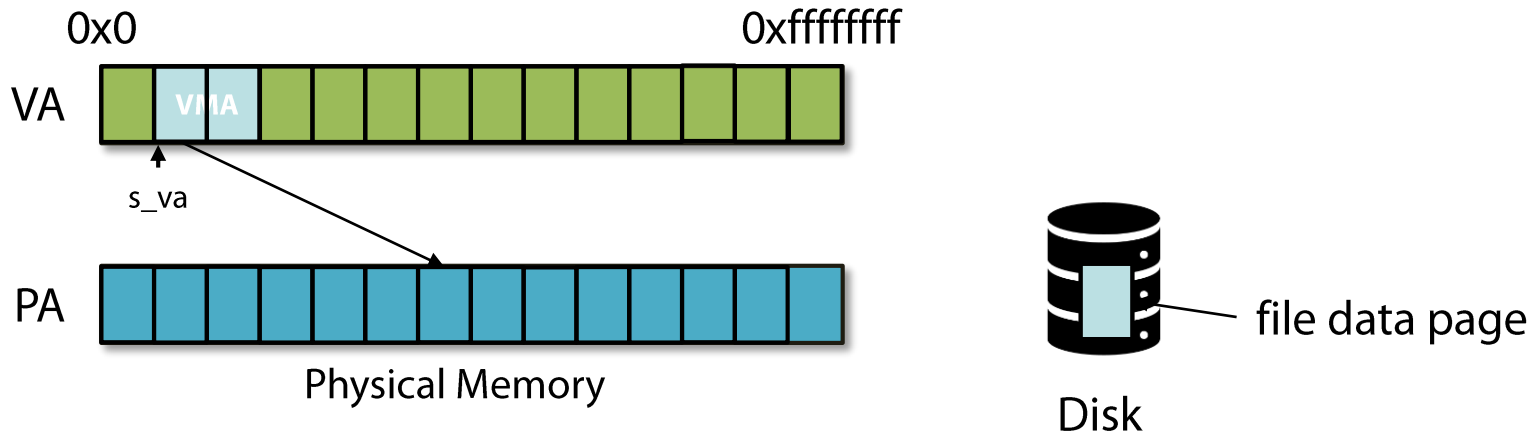
- ◆ Μετάφρασεις εικονικών διευθύνσεων σε φυσικές
 - ➔ Πίνακες σελίδων

Εικονική Μνήμη Επισκόπηση



- ◆ Μετάφρασεις εικονικών διευθύνσεων σε φυσικές
 - ➔ Πίνακες σελίδων
 - x86: πολυεπίπεδοι (δενδρική δομή radix tree)
 - ➔ Τους θέτει/διατηρεί το ΛΣ, τους διαβάζει/περπατάει το υλικό για να κάνει μεταφράσεις (MMU – page walk)

File mappings (Απεικονίσεις αρχείων)



- ◆ Εκτός από απεικονίσεις σωρού (heap buffers), η εικονική μνήμη μπορεί να χρησιμοποιηθεί και για την προσπέλαση αρχείων
 - ➔ `mmap()` αντί για `read()/write()` syscalls
- ◆ Το ΛΣ μεταφέρει σελίδες του αρχείου στη κύρια μνήμη

Σύνοψη



- ◆ Μηχανισμοί εικονικής μνήμης (virtual memory)
- ◆ Ζητούμενο 1: Κλήσεις συστήματος
 - ➔ Δημιουργία/διαχείριση απεικονίσεων (mmap() etc)
 - ➔ Εξέταση της απεικόνισης στη φυσική μνήμη
- ◆ Ζητούμενο 2: Parallel Mandelbrot με τη χρήση διεργασιών
 - ➔ Υλοποίηση σημαφόρων με διαμοιραζόμενη μνήμη
 - ➔ Υλοποίηση χωρίς συγχρονισμό με τη χρήση διαμοιραζόμενης μνήμης

Κλήσεις Συστήματος

- ◆ `mmap()`: δημιουργία μιας νέας απεικόνιση (mapping) στον χώρο εικονικών διευθύνσεων μιας διεργασίας

```
void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
```

- `prot`: δικαιώματα πρόσβασης (`PROT_READ/PROT_WRITE`)
 - `flags`:
 - Απεικονίσεις αρχείων ή σωρού (`MAP_ANONYMOUS`)
 - Ιδιωτικές ή διαμοιραζόμενες απεικονίσεις (`MAP_PRIVATE/MAP_SHARED`)
 - κ.α
 - `fd`: ο καταχωρητής αρχείου προς απεικόνιση
- ◆ `munmap()`: καταστροφή μιας απεικόνισης (ή εύρους της)
 - ◆ `mprotect()`: αλλαγή των δικαιωμάτων πρόσβασης μιας απεικόνισης (ή εύρους της)

Θα πρέπει να ανατρέξετε στο documentation (man pages) για μελέτη των ορισμάτων και δυνατοτήτων των syscalls

Z1: Δομή της άσκησης

- ◆ Καλείστε να συμπληρώσετε σε βήματα απλά μπλοκ κώδικα με απλή χρήση κλήσεων συστήματος

```
/*  
 * Step 1: Print the virtual address space layout of this process.  
 */  
printf(RED "\nStep 1: Print the virtual address space map of this "  
        "process [%d].\n" RESET, mypid);  
press_enter();  
/*  
 * TODO: Write your code here to complete Step 1.  
 */
```

Z1: Δομή της άσκησης

- ◆ Καλείστε να συμπληρώσετε σε βήματα απλά μπλοκ κώδικα με απλή χρήση κλήσεων συστήματος
 - ➔ Από το βήμα 7 καλείται η `fork()`

Father:

```
/*
 * Step 7: Print parent's and child's maps. What do you see?
 * Step 7 - Parent
 */
printf(RED "\nStep 7: Print parent's and child's map.\n" RESET);
press_enter();

/*
 * TODO: Write your code here to complete
 * parent's part of Step 7.
 */

if (-1 == kill(child_pid, SIGCONT))
    die("kill");
if (-1 == waitpid(child_pid, &status, WUNTRACED))
    die("waitpid");
```

Child:

```
/*
 * Step 7 - Child
 */
if (0 != raise(SIGSTOP))
    die("raise(SIGSTOP)");
/*
 * TODO: Write your code here to
 * complete child's part of Step 7.
 */
```

Z1: Δομή της άσκησης

- ◆ Καλείστε να συμπληρώσετε σε βήματα απλά μπλοκ κώδικα με απλή χρήση κλήσεων συστήματος
 - ➔ Από το βήμα 7 καλείται η `fork()`
 - ➔ Κομμάτια του δοσμένου κώδικα αφορούν στο συγχρονισμό των διεργασιών και δεν σας απασχολούν

```
/*  
 * Step 7: Print parent's and child's maps. What do you see?  
 * Step 7 - Parent  
 */  
printf(RED "\nStep 7: Print parent's and child's map.\n" RESET);  
press_enter();  
  
/*  
 * TODO: Write your code here to complete  
 * parent's part of Step 7.  
 */  
  
if (-1 == kill(child_pid, SIGCONT))  
    die("kill");  
if (-1 == waitpid(child_pid, &status, WUNTRACED))  
    die("waitpid");
```

```
/*  
 * Step 7 - Child  
 */  
if (0 != raise(SIGSTOP))  
    die("raise(SIGSTOP)");  
/*  
 * TODO: Write your code here to  
 * complete child's part of Step 7.  
 */
```

Z1: Δομή της άσκησης

- ◆ Καλείστε να συμπληρώσετε σε βήματα απλά μπλοκ κώδικα με απλή χρήση κλήσεων συστήματος
 - ➔ Από το βήμα 7 καλείται η `fork()`
 - ➔ Κομμάτια του δοσμένου κώδικα αφορούν στο συγχρονισμό των διεργασιών και δεν σας απασχολούν
- ◆ Για την παρατήρηση, μελέτη και κατανόηση των βασικών μηχανισμών διαχείρισης μνήμης → βοηθητικές συναρτήσεις
 - ➔ Χρήση τους σε κάποια βήματα
 - ➔ Ερωτήσεις με βάση τις παρατηρήσεις σας

(όλες οι απαντήσεις → θεωρία του μαθήματος)

Z1: Βοηθητικές συναρτήσεις

- ◆ `show_maps()`: τύπωμα χάρτη μνήμης της τρέχουσας διεργασίας
 - ➔ χάρτης μνήμης: όλες οι απεικονίσεις
 - ➔ `/proc/self/maps` interface*

VMA

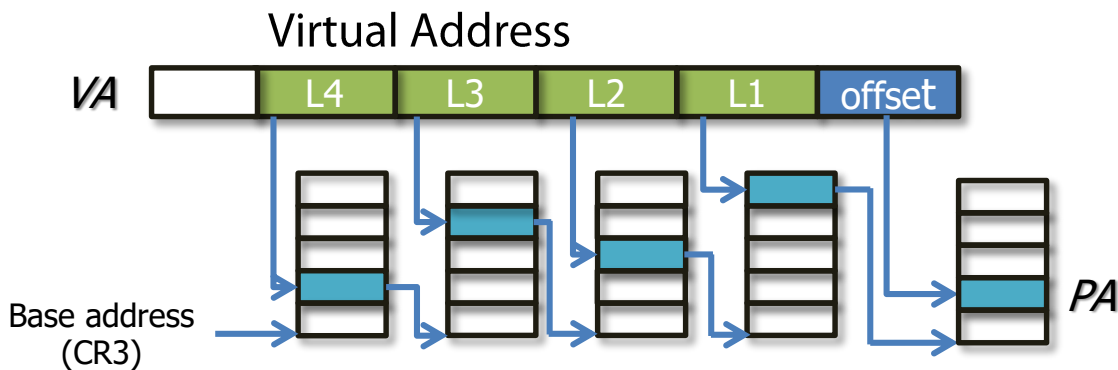
| | |
|--|--|
| 55d98a74b000-55d98a84f000 r-xp 00000000 103:07 8668845 | /bin/bash |
| 55d98aa4e000-55d98aa52000 r--p 00103000 103:07 8668845 | /bin/bash |
| 55d98aa52000-55d98aa5b000 rw-p 00107000 103:07 8668845 | /bin/bash |
| 55d98aa5b000-55d98aa65000 r-v-p 00000000 00:00 0 | |
| 55d98b32e000-55d98b3a8000 rw-p 00000000 00:00 0 | [heap] |
| 7fc01fc5d000-7fc01fc68000 r-xp 00000000 103:07 4981056 | /lib/x86_64-linux-gnu/libnss_files-2.27.so |
| 7fc01fc68000-7fc01fe67000 ---p 0000b000 103:07 4981056 | /lib/x86_64-linux-gnu/libnss_files-2.27.so |
| 7fc01fe67000-7fc01fe68000 r--p 0000a000 103:07 4981056 | /lib/x86_64-linux-gnu/libnss_files-2.27.so |
| 7fc01fe68000-7fc01fe69000 rw-p 0000b000 103:07 4981056 | /lib/x86_64-linux-gnu/libnss_files-2.27.so |
| 7fc01fe69000-7fc01fe6f000 rw-p 00000000 00:00 0 | |
| 7fc01fe6f000-7fc01fe86000 r-xp 00000000 103:07 4981053 | /lib/x86_64-linux-gnu/libnsl-2.27.so |
| 7fc01fe86000-7fc020085000 ---p 00017000 103:07 4981053 | /lib/x86_64-linux-gnu/libnsl-2.27.so |
| 7fc020085000-7fc020086000 r--p 00016000 103:07 4981053 | /lib/x86_64-linux-gnu/libnsl-2.27.so |
| 7fc020086000-7fc020087000 rw-p 00017000 103:07 4981053 | /lib/x86_64-linux-gnu/libnsl-2.27.so |

- ◆ `show_va_info(u64 va)`: τυπωμα των πληροφοριών μόνο για mapping

Μελετήστε τι σημαίνουν τα διάφορα πεδία που τυπώνονται

Z1: Βοηθητικές συναρτήσεις

- ◆ `uint64_t get_physical_address(uint64_t va)`: Επιστρέφει τη φυσική διεύθυνση (κύρια μνήμη) όπου απεικονίζεται η εικονική `va`.
 - ➔ `/proc/self/pagemap` interface*



Περπατάει τα page tables και βρίσκει τη μετάφραση αν υπάρχει

Σύνοψη

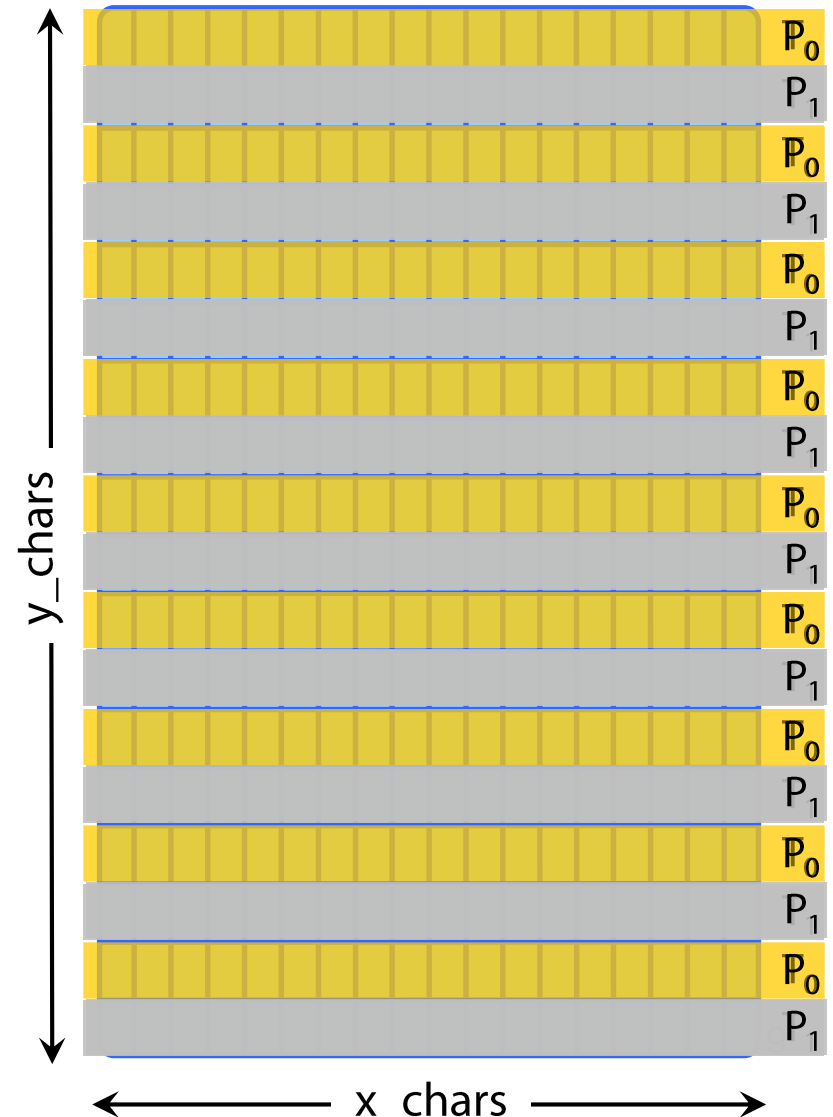


- ◆ Μηχανισμοί εικονικής μνήμης (virtual memory)
- ◆ Ζητούμενο 1: Κλήσεις συστήματος
 - ➔ Δημιουργία/διαχείριση απεικονίσεων (mmap() etc)
 - ➔ Εξέταση της απεικόνισης στη φυσική μνήμη
- ◆ Ζητούμενο 2: Parallel Mandelbrot με τη χρήση διεργασιών
 - ➔ Υλοποίηση σημαφόρων με διαμοιραζόμενη μνήμη
 - ➔ Υλοποίηση χωρίς συγχρονισμό με τη χρήση διαμοιραζόμενης μνήμης

Z2: Παραλληλοποίηση Mandelbrot με διεργασίες

- ◆ Με νήματα (άσκηση 3)
 - ◆ Νήμα i : γραμμές $i, i+N$ κτλ
- ◆ Με διεργασίες? (παρούσα)
 - ◆ `fork()` αντι για `pthread`
 - ◆ Αλλάζει η κατανομή?

Συγχρονισμός;



Z2.1: Σημαφόροι σε διαμοιραζόμενη μνήμη

- ◆ Πως επηρεάζεται ο μηχανισμός συγχρονισμού?
 - σημαφόροι → μοιραζόμενες μεταβλητές
 - fork() + δεσμευμένη μνήμη?
- ◆ Βασική διαφορά διεργασιών και νημάτων
 - νήματα → κοινή μνήμη
 - διεργασίες → διακριτή μνήμη
- ◆ Δια-διεργασιακός συγχρονισμός με σημαφόρους πάνω από διαμοιραζόμενη μνήμη
 - Συνεχίζετε να χρησιμοποιείτε τις ίδιες συναρτήσεις (sem_init, post, wait)
 - Πρέπει να ορίσετε/δεσμεύσετε τους σημαφόρους σε διαμοιραζόμενη μνήμη

Z2.2: Υλοποίηση χωρίς σηματοφόρους

- ◆ Οι διεργασίες αντί να τυπώνουν στην οθόνη γράφουν σε ένα διαμοιραζόμενο (προσπελάσιμο από όλες) buffer
 - ➔ buffer size: `y_char X x_char`
- ◆ Χρειάζεται πλέον συγχρονισμός?
 - ➔ Η main διεργασία τυπώνει τα περιεχόμενα του buffer
 - ➔ Πότε?

Z2: Σημαντικές συναρτήσεις

- ◆ `void* create_shared_memory_area()`: Δεσμεύει μνήμη προσπελάσιμη από όλες τις διεργασίες
 - ➔ Σας δίνεται σκελετός τον οποίο και πρέπει να συμπληρώσετε

Παράδειγμα χρήσης

```
void main () {  
    int *array = create_shared_memory_area(sizeof(int) * n);  
    for (i = 0; i < n; ++i)  
        array[i] = i;  
}
```

- ◆ `void destroy_shared_memory_area()`: Καταστρέφει διαμοιραζόμενες απεικονίσεις (shared mappings)
 - ➔ Σας δίνεται έτοιμη