Systemy operacyjne

Wykład 9: Zarządzanie pamięcią wirtualną

Unix: wywołania systemowe (1)

`sysconf(_SC_PAGESIZE)`	Podaje rozmiar strony, to samo co `getpagesize()`.
`mmap(addr, length, prot, flags, fd, offset)`	Odwzoruj **`length`** bajtów zasobu **`fd`** od pozycji **`offset`** lub pamięci anonimowej pod **`addr`**. Uprawnienia obszaru to **`prot`**.
`munmap(addr, length)`	Usuń fragment lub całość odwzorowania.
`mprotect(addr, length, prot)`	Zmień uprawnienia wybranego obszaru.
`msync(addr, length, flags)`	Synchronizuj odwzorowanie z obiektem. **`flags`** ∈ {`MS_ASYNC`, `MS_SYNC`, `MS_INVALIDATE`}

- **`lenght`** i **`offset`** podzielne przez rozmiar strony
- **`addr`** jest wskazówką dla **`mmap`** chyba, że **`MAP_FIXED`** ∈ **`flags`**
- **`prot`** złożeniem flag **`PROT_EXEC`**, **`PROT_READ`**, **`PROT_WRITE`**
- mapowanie może być prywatne (**`MAP_SHARED`**) lub dzielone (**`MAP_PRIVATE`**)
- pamięć anonimową przydzielamy flagą **`MAP_ANONYMOUS`**
- obszar może automatycznie rosnąć w dół **`MAP_GROWSDOWN`** (np. stos)
- **`MS_INVALIDATE`** unieważnia zawartość pozostałych odwzorowań zasobu

Unix: wywołania systemowe (2)

`mlock(addr, length)`	Przypnij strony do pamięci RAM.
`munlock(addr, length)`	Odepnij strony od pamięci RAM.
`mincore(addr, length, vec)`	Sprawdź czy strony są w rdzeniu (pradawne określenie RAM).
`madvise(addr, length, advice)`	Poinformuj system o zamiarach w stosunku do obszaru. **`flags`** ∈ {`MADV_NORMAL`, `MADV_RANDOM`, `MADV_SEQUENTIAL`, `MADV_WILLNEED`, `MADV_DONTNEED`}

- **`vec`** to adres miejsca na wektor bitów per strona
- przypięte strony → zabraniamy wymiany do pamięci drugorzędnej
- **`MADV_SEQUENTIAL`** → strony będą sprowadzane sekwencyjne, możesz włączyć gorliwe sprowadzanie (ang. *read-ahead*)
- **`MADV_WILLNEED`** → niedługo będę potrzebować, możesz asynchroniczne sprowadzić do pamięci RAM
- dużo flag *Linux-specific*, np. `MADV_MERGEABLE` (przydatne dla monitorów maszyn wirtualnych)

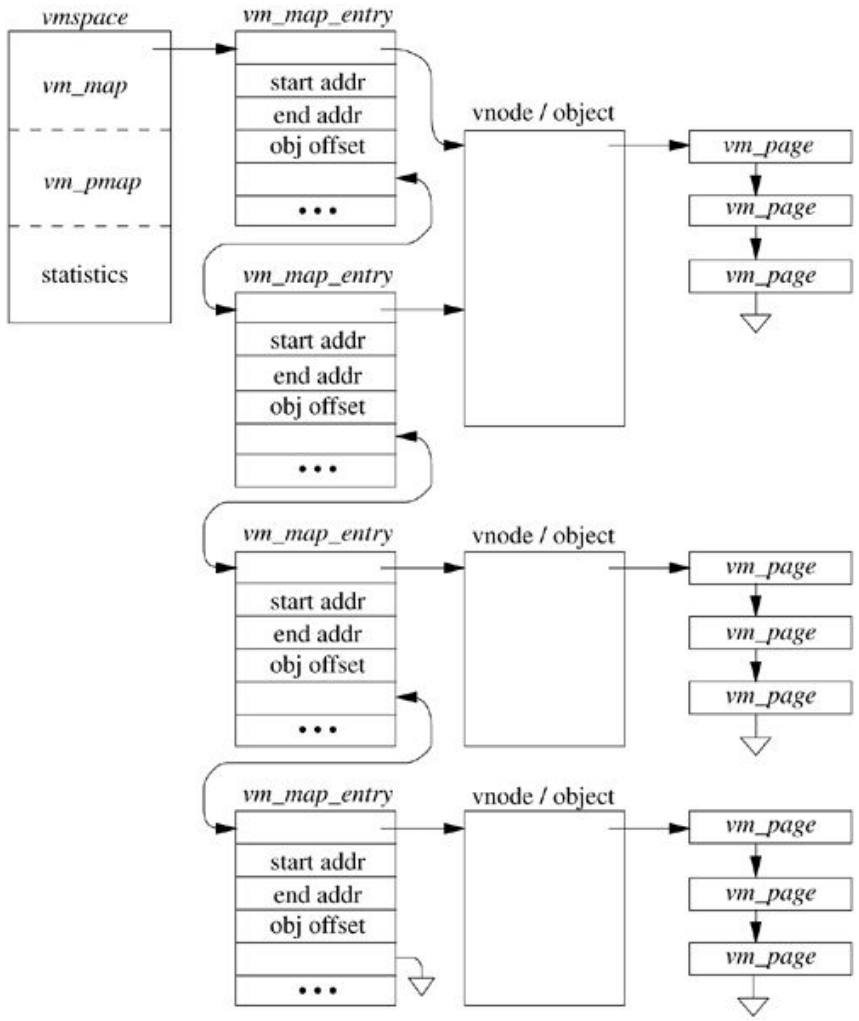
FreeBSD: Zarządzanie przestrzenią adresową

`vmspace` przechowuje tablicę stron (`pmap`), statystyki, wskaźniki do segmentów `text`, `data`, `bss`, oraz listę opisów obszarów adresów wirtualnych → **`vm_map_entry`**

`vm_object` dostarcza stron **`vm_page`**, które widać w danym przedziale adresów wirtualnych

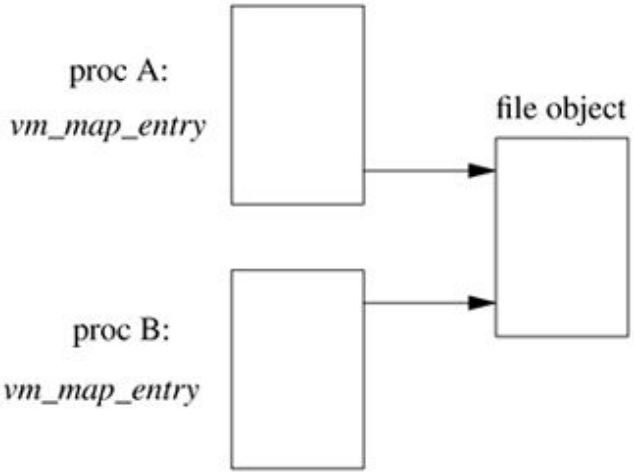
Linuksowy opis przestrzeni adresowej: **`cat /proc/$pid/maps`**

(`start`, `end`, `prot`, `offset`, `dev`, `inode`, `path`)



FreeBSD: Obiekt i procedury stronicujące

Operation	Description
pgo_init()	initialize pager
pgo_alloc()	allocate pager
pgo_dealloc()	deallocate pager
pgo_getpages()	read page(s) from backing store
pgo_putpages()	write page(s) to backing store
pgo_haspage()	check whether backing store has a page
pgo_pageunswapped()	remove a page from backing store (swap pager only)

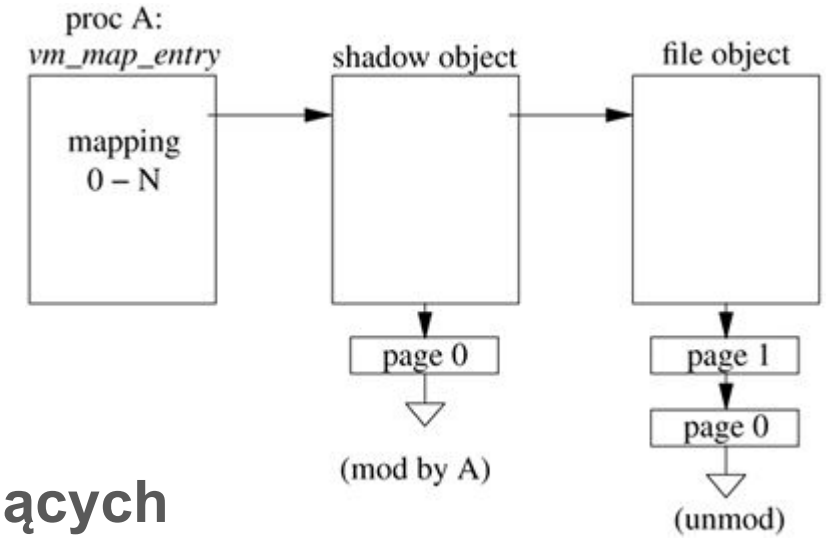


Z każdym obiektem skojarzona lista stron, procedury stronicujące, licznik referencji, itp. Obiekt może odpowiadać pamięci anonimowej, plikowi, urządzeniu. Można go też współdzielić między procesy!

`swap pager` → **`getpages`** zwraca wyzerowaną stronę anonimową
`vnode pager` → **`getpages`** przydziela stronę i ładuje do niej kawałek pliku

FreeBSD: Mapowanie prywatne plików

`MAP_PRIVATE` dla pliku tworzy odwzorowanie z kopią przy zapisie. Zmiany nie są zapisywane do pliku i nie są widziane w pozostałych procesach, które mapują ten zasób.



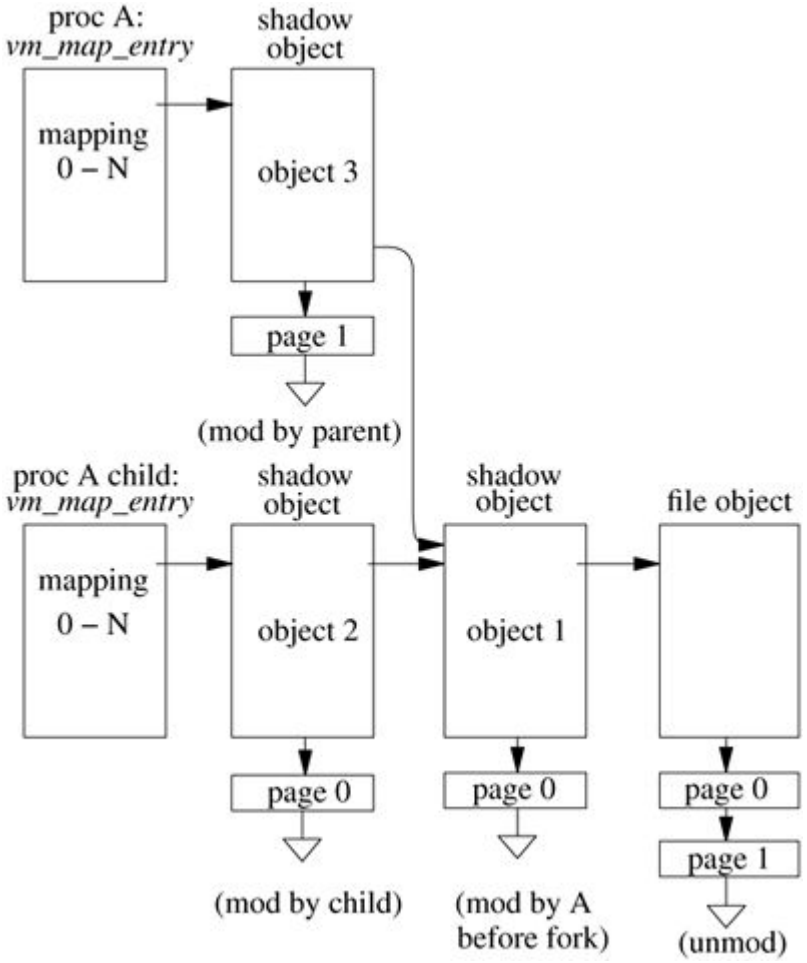
Potrzebujemy **obiektów przesłaniających** (ang. *shadow object*). Oryginalne strony są tylko do odczytu! Kiedy zapisujemy → błąd strony! Przydzielamy stronę anonimową, kopiujemy zawartość oryginału (ang. *copy-on-write*), podczepiamy do obiektu przesłaniającego i wpisujemy do tablicy stron (pmap_enter) w miejsce oryginału.

FreeBSD: Klonowanie przestrzeni adresowych

W momencie klonowania (**`fork`**) tworzymy rodzicowi i dziecku obiekty przesłaniające oryginalną zawartość przestrzeni adresowej, a oryginalne strony ustawiamy tylko do odczytu.

Q: Co jeśli proces **`A`** się zakończy?
A: Zwalniamy obiekt 3 i jego stronę 1. Zostaje nam ciąg obiektów cieni, które trzeba złożyć (ang. *collapse*).

Q: Co jeśli obiekt 1 zawiera stronę 1?
A: Składamy obiekty w kolejności 2 → 1, i przenosimy do 2 tylko najświeższe kopie.



FreeBSD: Obsługa błędu strony

Obsługujemy wyjątek CPU wstrzymując wątek procesu, odczytujemy rejestry sprzętowe, znajdujemy bieżącą przestrzeń adresową i wołamy:

```
int vm_fault(vm_map_t map, vm_offset_t vaddr, vm_prot_t type)
```

Uproszczona wersja bez optymalizacji i blokad (FreeBSD, §6.11):

1. Przeszukaj listę w poszukiwaniu **`vm_map_entry`**, do którego przynależy **`vaddr`**. Nie → wyślij **`SIGSEGV`** (**`SEGV_MAPERR`**)!
2. Czy obszar posiada stronę, na której leży **`vaddr`**?
   Nie → zawołaj **`pgo_getpages(object, page)`** i podczep stronę!
3. Uprawnienia się nie zgadzają?
   a. `shadow object` → znajdź stronę głębiej, skopiuj i podczep!
   b. `pager` → wyślij **`SIGSEGV`** (**`SEGV_ACCERR`**)

FreeBSD: Klasy stron w jądrze

Podejrzyjmy statystyki pamięci wirtualnej: **`vmstat -s`**

- **`WIRED`** strony przyczepione do pamięci operacyjnej, używane przez jądro lub przypięte wywołaniem **`mlock`**
- **`ACTIVE`** prawdopodobnie należą do zbiorów roboczych procesów, jądro bada użycie tych stron i przenosi do listy **`INACTIVE`**
- **`INACTIVE`** nieużywane i potencjalnie brudne, po wyczyszczeniu trafiają do listy **`CACHE`**, przy błędzie strony wracają do **`ACTIVE`**
- **`CACHE`** nieużywane i czyste, licznik referencji ustawiony na zero
- **`FREE`** strony gotowe do przydzielenia, być może wyzerowane

Demon stronicowania dąży do tego, by na liście **`FREE`** + **`CACHE`** oraz **`INACTIVE`** znajdowało się odpowiednia ilość (procentowo) pamięci.

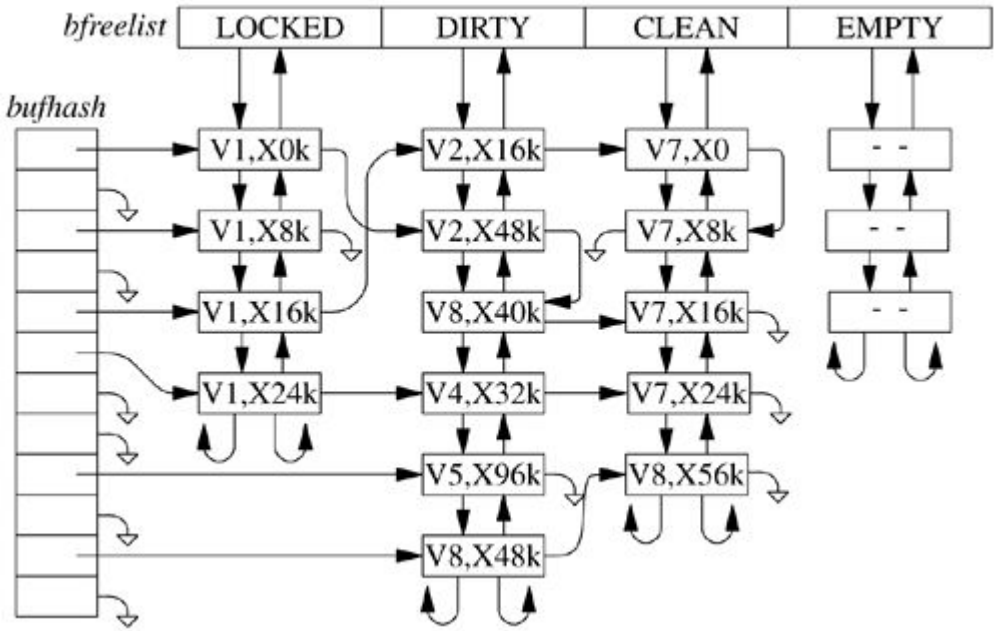
FreeBSD: Buforowanie stron

System posiada osobne bufory dla stron anonimowych oraz stron należących do plików (w tym urządzeń blokowych). Strony i bloki dyskowe są traktowane tak samo → page cache.

Q: Jak wyznaczyć położenie strony należącej do pliku?
A: Potrzebujemy identyfikator niezależny od systemu plików (vnode) i pozycję strony w pliku.

`bufhash` kubełki adresowane parą (**`vnode`**, **`offset`**)

`LOCKED` → na zawartości wykonywane operacje wej.-wyj.



Przełączanie przestrzeni adresowych

Zmieniamy wskaźnik na tablicę stron – to wszystko? **NIE!**

W **TLB** wpisy ze starej tablicy stron… Pod tymi adresami w innym procesie jest coś innego → opróżnić? **NIEDOBRZE!**

Co z pamięcią podręczną? Tagowana adresami fizycznymi → ok!
Tagowana wirtualnymi (często dla L1) → opróżnić cache? **BOLI!**

Sprzęt oferuje pulę (2^n gdzie n małe) **identyfikatorów przestrzeni adresowych** (**`ASID`**). Każdy wpis w TLB i cache ma pole ASID, które sprzęt porównuje z zawartością uprzywilejowanego rejestru.

`Mach3`: Zarządzanie tablicą stron

Implementacja translacji adresów i tablicy stron mogą się znacząco różnić między architekturami (`Intel` vs. `PowerPC` vs. `MIPS`).

Potrzebujemy abstrakcyjnego interfejsu do zarządzania translacją adresów, uprawnieniami stron, bitami monitorowania dostępu, itp.

Moduł pmap (ang. *physical map*) zaprojektowany dla jądra Mach! Początek lat ‘90. Używany obiecnie w systemach BSD i MacOS X.

Zarządza pamięcią niezbędną do zbudowania struktur danych dla sprzętowego lub programowego przeglądania tablicy stron. Emuluje funkcje niedostępne w sprzęcie. Zna format wpisów tablicy stron. Zarządza sprzętowymi numerami przestrzeni adresowych (ASID). Unieważnia wpisy w TLB i pamięci podręcznej.

pmap: relacja między ramkami, a stronami

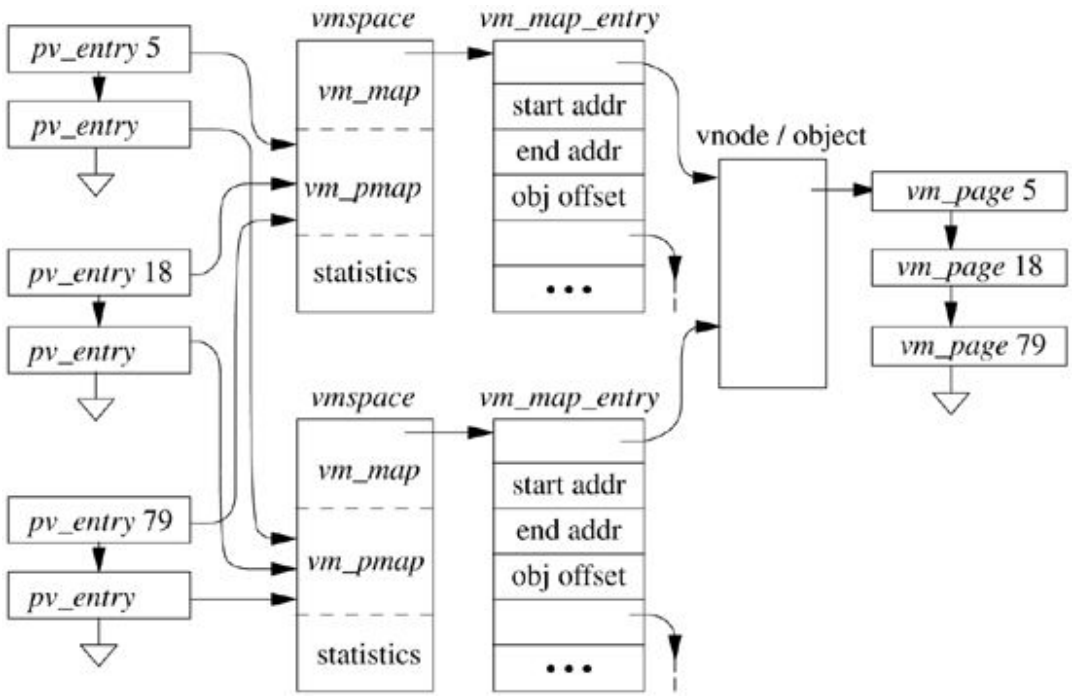
UWAGA! `vm_page` to opis ramki → nie ma adresu wirtualnego!

Jak znaleźć **vmspace** do których została podczepiona ramka?

md_page czyli część **vm_page** zależna od architektury sprzętowej przechowuje głowę listy **pv_entry**

```
struct md_page {
    TAILQ_HEAD(, pv_entry) pv_list;
};

struct pv_entry {
  pmap_t pv_pmap;
  vm_offset_t pv_va;
  TAILQ_ENTRY(pv_entry) pv_link;
};
```



pmap: podczepianie ramek i zmiana uprawnień

```
int pmap_enter(pmap_t, vm_addr_t va, vm_page_t pg, vm_prot_t, ...);
                    void pmap_zero_page(vm_page_t);
                void pmap_copy_page(vm_page_t, vm_page_t);
        void pmap_protect(pmap_t, vm_addr_t, vm_addr_t, vm_prot_t);
```

Możliwe uprawnienia: **VM_PROT_{NONE,READ,WRITE,EXECUTE,COPY}**

pmap_enter wprowadza mapowanie ramki pg pod adresem va z odpowiednimi prawami dostępu, wołane przy błędzie strony

pmap_zero_page zanim udostępnimy anonimową stronę, należy ją podpiąć do adresów wirtualnych jądra (KVA) i wyzerować
pmap_copy_page j.w, używane przy klonowaniu strony przy błędzie strony spowodowanym działaniem mechanizmu *copy-on-write*

pmap_protect zmienia uprawnienia dostępu do przedziału stron

pmap: odczepianie ramek

```
void pmap_remove(pmap_t, vm_addr_t, vm_addr_t);
void pmap_remove_all(vm_page_t);
void pmap_remove_write(vm_page_t);
```

pmap_remove odczepia strony z podanego przedziału,
np. w wyniku wywołania systemowego **munmap**

pmap_remove_all przegląda listę **pv_entry** związanych z daną ramką i odczepia je od odpowiednich przestrzeni wirtualnych, używane przez algorytm wymiany ramek

pmap_remove_write używane wewnętrznie przez podsystem pamięci wirtualnej do skonfigurowania ramki do użycia jako *copy-on-write* przy wywołaniu fork

pmap: bity dostępu i programowa translacja adresu

```
boolean_t pmap_is_modified(vm_page_t);
  void pmap_clear_modify(vm_page_t);
  int pmap_ts_referenced(vm_page_t);
```

pmap_is_modified sprawdza bit `modified` dla danej ramki
pmap_clear_modified czyści bit `modified` dla danej ramki
pmap_ts_referenced zwraca wartość licznika dostępów (jeśli sprzęt to udostępnia) dla danej ramki i czyści go

```
vm_paddr_t pmap_extract(pmap_t, vm_addr_t);
```

pmap_extract tłumaczy podany adres wirtualny na fizyczny według bieżącej zawartości sprzętowej tablicy stron

pmap: przełączanie przestrzeni adresowych

```
void pmap_activate(thread_t *);
```

Wołane jeśli jądro chce coś skopiować do przestrzeni adresowej wątku lub przełączyć na jego kontekst.

Sprzęt oferuje identyfikatory przestrzeni adresowych?

Tak → jądro utrzymuje listę aktywnych procesów i przypisuje im dostępne ASID. Za dużo procesów? Któryś wypada ze zbioru i jest zastępowany! Czyścimy wpisy TLB i linie cache z ustalonym ASID.

Nie → czyścimy wszystko jak idzie.

Pytania?