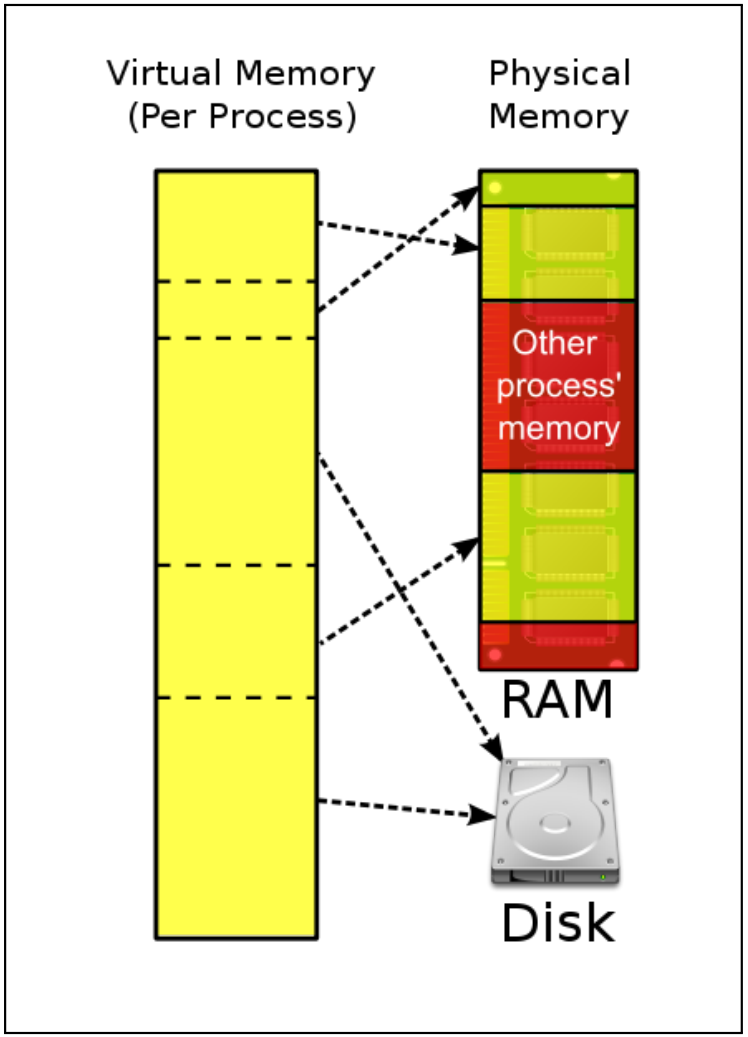
Laborator 7

Memoria virtuală

Sisteme de Operare

31 martie - 6 aprilie 2011

- Mecanism folosit implicit..
  - de către nucleul sistemului de operare pentru a implementa o politică eficientă de gestiune a memoriei
  - ce astfel de optimizări cunoașteți?



imagine preluată de pe wikipedia.org

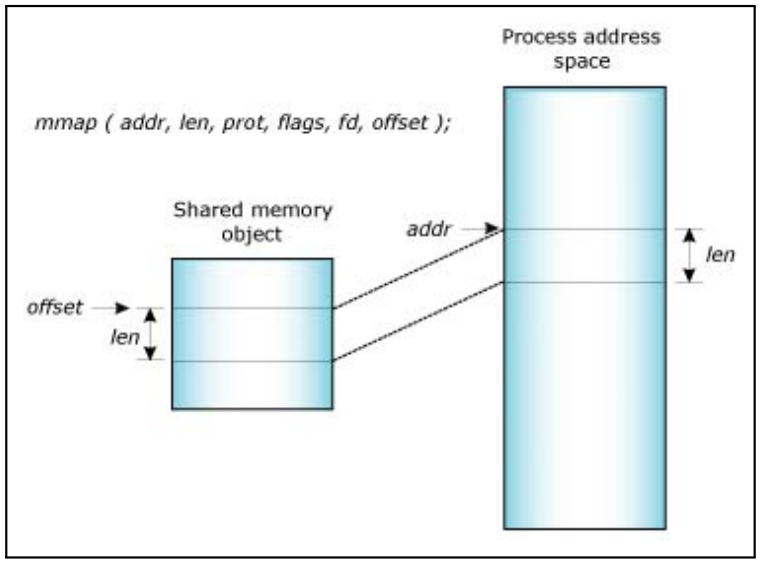
Memoria virtuală(2)

- .. dar și explicit, pentru a mapa în spațiul de adresă al unui proces:
  - fișiere
  - memorie
  - dispozitive
- Mapare fișiere
  - memorie partajată
  - paginare la cerere
  - biblioteci partajate
- Mapare memorie
  - pentru alocarea unei cantități mari de memorie
- Mapare dispozitive
  - acces direct la memoria dispozitivului
  - când ar putea fi necesar?

Mapare fișiere și memorie - Linux(1)

- Accesare fișier similar cu un vector
- Maparile pot depăși dimensiunea memoriei fizice
- Nu pot fi mapate dispozitive cu acces secvențial (socket-uri, pipe-uri)
- Unitate: pagina (număr întreg, alinieri)
- Familia de funcții mmap(2)

- mmap/munmap



- start poate fi NULL
- prot: PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
- flags: MAP_PRIVATE, MAP_SHARED, MAP_FIXED, MAP_LOCKED, MAP_ANONYMOUS (pt mapare memorie)
- mapare memorie: ignoră fd și offset
- msync - sincronizare explicită fișier cu maparea din memorie

Mapare fișiere - Windows

- CreateFileMapping/OpenFileMapping
  - primește HANDLE fișier
  - tip mapare: PAGE_READONLY, PAGE_READWRITE, PAGE_WRITECOPY
- MapViewOfFile
  - primește HANDLE FileMapping
  - mod acces: FILE_MAP_READ, FILE_MAP_WRITE, FILE_MAP_COPY
- UnmapMapViewOfFile

Mapare memorie - Windows

- VirtualAlloc/VirtualAllocEx
  - tip operație: MEM_RESERVE, MEM_COMMIT, MEM_RESET
  - start poate fi NULL; multiplu de 4KB pentru alocare și 64KB pentru rezervare
- VirtualFree/VirtualFreeEx
  - tip operație: MEM_DECOMMIT, MEM_RELEASE
- Interogarea zonelor mapate VirtualQuery/VirtualQueryEx
  - adresa de start a zonei, protecție, dimensiune
  - struct _MEMORY_BASIC_INFORMATION

Excepții.Schimbarea protecției unei zone mapate

- Accese la memorie nonconforme cu drepturile
  - Linux - generează semnale SIGBUS, SIGSEGV
    - sigaction, siginfo_t
  - Windows - generează excepții
    - AddVectoredExceptionHandler, VectoredHandler
- Linux - mprotect
  - acces: PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
  - adresa multiplu de dimensiunea unei pagini
- Windows - VirtualProtect/VirtualProtecTEx
  - pt regiuni alocate cu VirtualAlloc/VirtualAllocEx folosind MEM_RESERVE

Blocarea paginării

- Utilă pentru procese care trebuie să execute anumite acțiuni la momente de timp bine determinate
- Nu se va mai face swapout - ulterioare nu mai produc page fault
- Linux
  - mlock
  - mlockall
    - flags: MCL_CURRENT, MCL_FUTURE
  - munlock/munlockall
- Windows
  - VirtualLock/VirtualLockEx
    - rezultat: TRUE succes, FALSE altfel
  - VirtualUnlock/VirtualUnlockEx

Întrebări

- Câte page fault-uri se pot genera în urma execuției următorului cod:
  char *a = malloc(5000);
  memset(a, 0, 5000);
- De ce este apelul malloc mai rapid decât calloc?
- Pot două procese să partajeze o pagină virtuală?
- Dacă un access la memorie durează 50 ns, cât durează execuția următorului cod:
  int a[1030];
  a[0] = a[1029] = 42;
  Presupunem că avem un sistem cu paginare simplă.