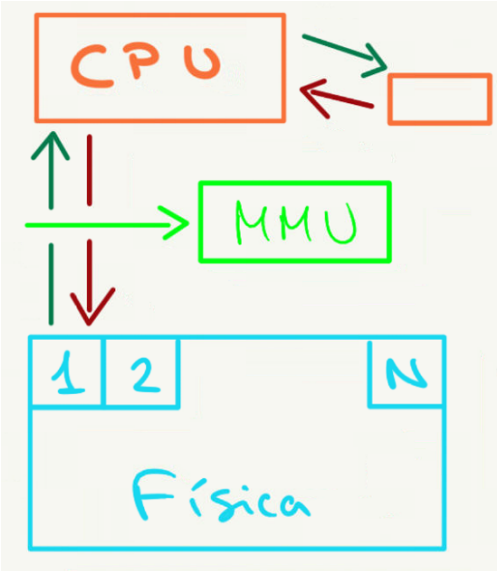
TEMA 3: MEMORIA

Assign	
Status	

La CPU només pot accedir directament a memòria física (RAM) i registres, per tant, les instruccions, les quals inicialment és troben a la memòria ROM, s'han de carregar a memòria física abans de poder referenciar-se. Trobem dos tipus de memòria:



- Adreça lògica → Referencia emesa per la CPU.
- Adreça física → Posició ocupada en memòria.

Si el sistema operatiu i el hardware permeten la traducció d'adreces, no tenen perque coincidir. Mitjançant el Kernel i la MMU (Memory Managment Unit), un element de hardware, cada accés a memòria és tradueix. Contrariament, si no disposem de

cap mecanisme de traducció, les adreces físiques i lògiques són equivalents.

L'espai de direccions d'un procès són el conjunt d'adreces que pot emetre la CPU.

Per una altra part, el procés executable té espai de direccions físiques i lògiques. El físic, conté les adreces que té associades, mentre que el lògic conté les adreces a les que el Kernel li ha permès accedir.

Correspondencia fisicològica

La correspondencia entre adreces físiques i lògiques pot ser o bé fixa o bé a través de traducció. La fixa es la més senzilla, ja que estipula que l'adreça de la memòria física és la mateixa que la de l'adreça lògica. Per l'altra banda, la traducció consisteix en traduir, mitjançant la MMU, la direcció a la que el Kernel ha assignat el programa. Finalment, el Kernel configura el linkatge d'adreces (Punters a la nova adreça).

Sistemes multiprogramats

Diversos programes poden estar carregats a la CPU simultaneament. Ens permet l'execució concurrent i simplifica els canvis de context.

> 💻 Un procés a CPU i N processos a memòria física. Al canviar de procés no es neccesari carregar de nou el procés.

El sistema operatiu ha de garantir la protecció de les dades. Per part de hardware la MMU detecta els accesos il·legals a memòria. A cada canvi de context el kernel ha d'actualitzar la MMU.

Serveis bàsics del S.O.

Format d'un executable

Els fitxers binaris contenen adreces lògiques i físiques. A la capçalera trobem: tipus, mida i posicions dins del binari. El format estandaritzat d'una arxiu binari es el ELF:

- `.text` → Codi
- `.data` → Dades globals inicialitzades
- `.debug` → Informació de debug

- `.bss` → Dades globals sense valor inicial
- `.init` → Codi de inicialització d'un procés
- `.comment` → Informació de control
- `.dynamic` → Informació d'enllaç dinàmic

Optimització sota demanda

Una rutina només es carrega quan s'ha d'utilitzar, això ens permet una major eficiencia de la memòria, ja que no es carregan funcions innecessaries. Per tal de comprovar si una rutina està carregada en memòria el sistema operatiu registra, en les seves estructures de dades, quines zones de memòria són valides i on llegir els continguts. Quan el procés accedeix a la adreça, la MMU genera una excepció per avisar al sistema operatiu que l'adreça no existeix, llavors el sistema operatiu comprova en les seves estructures de dades si el accés és valid, o no. Provoca la carrega i reactiva l'execució de la instrucció que ha provocat l'excepció.

Optimització de llibreries compartides

Els binaris dels programes, no contenen el contingut sencer d'una llibreria ja que ocuparia massa, contenen un punter al codi d'aquesta. Si tots els programes on afegim una llibreria tinguessin una còpia, tindriem molt espai ocupat per codi duplicat.

En cas d'actualització de la llibreria, també, no cal canviar tots els codis duplicats ja que només farà faltar canviar el codi on apunten tots els binaris que contenen la llibreria.

Per tant, els processos tenen, en aquests casos, memòria compartida. Aquesta memòria es de ús de lectura exclusiu, com és el cas de les llibreries.

Reservar o alliberar memòria dinàmica

Hi ha variables que no tenen una mida fixa, per tant, si fixem el tamany d'aquestes variables en el moment de compilació podem desaprofitar memòria, o pel contrari, trobar-nos en accesos erronis de memòria i rebre el signal `SIGSEGV` . El sistema operatiu ofereix crides a sistema per reservar noves regions de memòria (memòria dinàmica). La memòria dinàmica s'emmatzema en la zona heap del espai d'adreçes lògiques.

Mitjançant la crida a sistema `sbrk( quantitatDeMemòria (int) )` podem modificar el límit del heap. El sistema operatiu no te conciencia de quines variables estan ubicades a

cada zona, simplement modifica la mida del heap. També trobem la instrucció `malloc` , la qual s'utilitza per demanar memòria. La instrucció complementaria al `malloc`, és `free` .

💻 malloc → Coloca els n bytes i retorna un punter a la memòria. La memòria no s'inicialitza. Si la mida és 0, llavors la funció ens retorna un punter NULL, o el valor d'un punter unic que després passarà a lliure.

💻 free → Allibera l'espai de memoria apuntat per ptr. Si el punter apunta a un lloc NULL, no es realitza cap operació.

💻 sbrk → S'utilitza per canviar dinàmicament la quantitat d'espai assignat per al segment de dades del procés de trucada. El canvi es realitza restablint el valor de ruptura del procés i assignant la quantitat d'espai adequat.

Assignació de memòria

L'assignació de memòria s'executa cada vegada que un procès necessita memòria física. Per exemple, en Linux, la creació d'un fill o la mutació d'un programa en necessiten.

1.- Per començar s'ha de seleccionar memòria física lliure i definir-la com a ocupada.

2.- Seguidament, s'ha de implementar la traducció amb MMU.

3.- Solucionar els possibles problemes de fragmentació.

Problemes de fragmentació

La fragmentació de memòria és el cas de memòria que està lliure pero no pot ser utilitzada per un procés. Podem dividir-la en fragmentació interna i externa. La interna és deguda a la reserva de memòria per un programa que no la necessita. No esta ocupada, pero esta reservada; per tant no pot ser utilitzada per un altre procès. El cas de la externa es produeix quan hi ha molts espais de memòria lliures pero no de forma continuada, per tant no ens serveix per assignar-la a un programa (És a

dir, hi ha 16 bytes lliures a `0x00FF`, pero no hi torna a haver cap espai lliure fins a `0x2000`)

Assignació contigua

Els espais d'adreces de memòria reservats son contigus, es a dir, estan seguits. Tot el procès ocupa una partició que es selecciona a l'inici de l'execució. És poc flexible.

Assignació no-contigua

Els espais d'adreces de memòria reservats no son contigus, es a dir, no estan seguits. Augmenta la flexibilitat, però també la complexitat del treball del S.O. i de la MMU.

L'assignacipot tenir un esquema basat en paginació. Els espais de memòria es divideixen en particions de tamany fixe anomenades pàgines. Per tant, es busquen, dins de cada pàgina, blocs lliures. Quan un procès acaba la seva execució, torna a marcar com a lliures els blocs de la seva pàgina.

Aquesta tecnica facilita la compartició de memòria.

Com hem vist abans, la MMU s'encarrega de comprovar els accessos a memòria. En el cas de la paginació, també ho farà. Treballa conjuntament amb el TLB (cache), on s'emmagatzema l'informació de les pagines actives:

MMU

Emmagatzema informació de traducció de adreces.

Te les adreces de totes les pagines dels programes actius.

TLB

Emmagatzema informació sobre les pàgines actives.

No conté totes les adreces ja que és una memòria més ràpida però més reduïda.

Quan es produeix un canvi a la MMU, s'ha d'actualitzar la TLB.

L'assignació també por prendre l'esquema de segmentació. Aquest divideix l'espai de memòria en el tipus de contingut. Es a dir, separa en segments el codi, la pila i les dades. Per accedir-hi per tant, només caldrà un punter al segment i moure'ns per dins del mateix.

Existeix, per últim, el cas de l'unió dels dos esquemes de assignació s'anomena segmentació paginada. Per al cas de processos grans, cada segment dels que acabem de parlar es separaria dins en pàgines.

Compartició de memòria

Podem especificar si compartir la memòria entre processos a nivell de pàgina o a nivell de segment. Per processos que executen el mateix codi no es neccesari múltiples copies a memòria física, només neccesitem accés a lectura.

El sistema operatiu ens proporciona crides a sistema perque un procés afegeixi zones de memòria en el seu espai lògic. La memòria compartida està com a mecanisme de comunicació entre processos. La resta de memòria de la que disposa un procés és privada.

Optimitzacions de còpia en escriptura COW.

L'objectiu és evitar inicialitzacions de memòria prematures. Si no hem de accedir ni modificar una zona, no cal que la reservem o dupliquem. Així estalviem temps d'execució i espai de memòria.

En el cas de la creació d'un procès fill (`fork`), evitem inicialitzar parts de memòria que siguin només de lectura, ja que no cal tenir-les carregades si no hem d'escriure.

Per implementar-lo, cal saber quan la memòria sí que serà necessaria, per a reservar-la. En quant es fa l'assignació, al S.O. es marca l'espai de memòria amb els permissos d'accés, i a la MMU amb accés únic de lectura. Per tant, si un procès intenta escriure a una zona de memòria, la MMU llença una excepció.

Memòria virtual

La memòria virtual és una extensió de l'idea de carrega sota demanda. Permet prendre i treure dades de memòria sota demanda. Pretén reduir la quantitat de memòria assignada a un procediment en execució.

Per optimitzar, existeix l'idea de intercanvi (`swapping`). Només cal tenir carregat el procés actiu (amb CPU assignada). En cas que el procès actiu necessiti més memòria, es pot expulsar altres processos temporalment (`swap out`).

Els processos en espera per tornar a ocupar la CPU, s'emmagatzemen en un emmagatzament secundari (`backing storage`).

Abans de reprende l'execució d'un dels processos en espera, s'ha de tornar a carregar en memòria, cosa que ralentitza l'execució.

En quant el sistema necessita lliberar blocs de les pàgines, es procedeix al reemplaçament de memòria: Es selecciona una pàgina, s'elimina la seva traducció, es guarda temporalment el seu contingut, i s'assigna aquesta pàgina ja buida al nou bloc que la necessita.

Aquest guardat temporal, es fa a l'area de Swap. La MMU no té traducció per tant si s'intenta accedir és genera una excepció. Es poden revertir els canvis anteriors mitjançant els passos inversos gràcies al S.O.

Gràcies al ús de memòria virtual podem tenir més espais lògics que memòria física. No obstant, les fallades de pàgina son molt costoses en temps.

L'algorisme de reemplaçament ha de minimitzar les fallades de pàgina, seleccionant pàgines que fa molt que no s'utilitzen (LRU) com a pàgina d'esborrat.

Memòria prefetch

L'objectiu es minimitzar el número de fallades de pagina, i es fa anticipant quines son les que es necessitaràn i carregant-les en memòria abans que les demani el procès. Té diferents parametres:

- Distancia de prefetch: Marca amb quina antelació volem anticipar la càrrega.
- Numero de pàgines a carregar: Marca quantes pàgines carregarem en memòria de forma anticipada.

Per tal de predir quines pàgines es voldràn fer servir, existeixen algorismes com "Strided" o "Secuencial".