

Esquema

Memoria del SO

Memoria → Niveles (Gestión)

De SO

- Asigna particiones a procesos (No crítica)
- Asignar a subsistemas del SO (Crítica)

De procesos

Gestión dinámica
(malloc, free...)
(Distribuidores)

→ Asignación → Memoria Kernel → Sist. amigo: grandes block contiguos
→ Mecanismo vmalloc() block no contiguos

Memoria usuario: mecanismos → Gestión → cap. direcciones
de procesos

Reclamación recuperar memoria ya no usada

Diferencias ⇒ Usuario vs Kernel

Petic. no urgentes

Urgentes

Proc. no confiable

Preparado para errores direcciónam.

Esp. direc./process

—

Puede añadir/sup. rangos de direcciones

APIs (Linux) → Int. de llamadas al sistema (User)
→ Int. intra-Kernel

Memoria de procesos

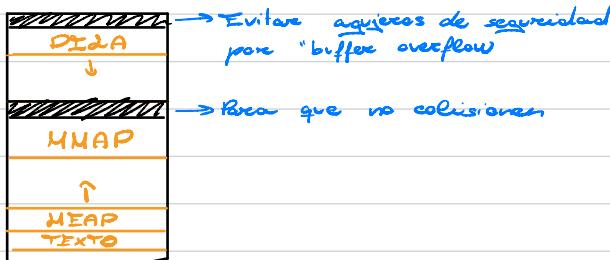
Paginación multí nivel

Pag. simple \Rightarrow TDP tam muy elevado \Rightarrow No permite que I muchos procesos

\rightarrow 2 niveles Tablas de 1º nivel direcciones de 2º nivel
que direccionan memoria

✓
no tienen pag.
están en mem. prin.

Espacio Virtual en x86



Regiones (Linux)

Def. Recurso para representar intervalos de direc. contiguas con la misma protección

Caract.

- Inicio
- longitud (tamaño de la página)
- Derechos de acceso

Áreas de memoria virtual

TP \Rightarrow Esp. de direcciones no muy grandes

Cada región \Rightarrow Estructura VM-área \Rightarrow solo perten. a 1 proc.
L \rightarrow información para traducción
de direcciones que la TP no puede

Espacio de direcciones de un proceso
(Representación)

Lista de VM-áreas
(Pocas VM-áreas)

Árbol rojo-negro
(Número elevado)

Elementos

- Rango de direcciones: inicio y fin región
- Indicadores VM: VM_READ, VM_WRITE, VM_EXEC
- Información de enlace: puntero a la lista de VM-áreas
- Operaciones: punteros a funciones / rutina para ^{y datos priv.}
- Info de archivo proyectado: si el VM-área es una proyección, puntero al archivo y desplazamiento

Operaciones: VM-áreas como objeto, con diferentes manejadores
L \rightarrow Cualquier objeto proyectado puede suministrarse operaciones

Llamadas al sistema: relacionadas con regiones

brkC \Rightarrow Heap

munmapC

execC

shmatC \Rightarrow Área región compartida

exitC

shmrdtC \Rightarrow Elimina "

forkC

mmapC

Memoria virtual

Separamos esp. de direcciones de la mem. física

Disco como extensión de la RAM

↳ Almacena páginas que no están en uso

↳ Conjunto residente (Pág. en disco)

Paginación por demanda

Ventajas:

- Reduce %/s
- - mem. en uso
- + rápida la respuesta
- + procesos en ejec.

¿Cuándo demanda?

↳ Ref. inválida \Rightarrow Abort.

Proc. referencia algo de la pág. \rightarrow Ref. válida \Rightarrow se trae a mem.

Falta de página:

Para paginación por demanda hace falta...

Hardware \Rightarrow Excepc. falta de pág.

Software \Rightarrow Rutina de excepc. falta de página

Para cada PTE:

Bit válido = 1 \Rightarrow Pág. en memoria

Bit válido = 0 \Rightarrow O ref. inválida o en disco

falta de página

Tratamiento:

Comprueba si el ref. válido \Rightarrow Obtiene marco vacío \Rightarrow Carga la pág.

Activa L1

cont. proceso

Bit de validación revisado

Puede que página esté en memoria pero bit=0.

Motivos:

- Simular en soft el bit de referencia
- copy-on-write: compartir páginas de lectura y duplicar para la escritura

Sustituciones de páginas

Objetivo: liberar marcos con páginas que no están en uso para usarlos

Algoritmos LRU → LRU

Windows → Kelley(LRU), aleatorio

OS x → Segunda oportunidad

→ LRU

Cost. de referencia: sec. de núm. de páginas referenciadas por un proceso en ese.

LRU elige la página cuyas referencias consec.

están más separadas en la cost. de referencia

→ las que menos frecuentemente se usan

Si hay más de una opción, se elige arbitrariamente

Es buen algoritmo para el princ. de localidad

Bit sucio: ¿Se ha modif. pág.? \rightarrow Si \Rightarrow Bit=1 \Rightarrow Hay que salvar
 \rightarrow No \Rightarrow Bit=0 \Rightarrow No hace falta

Mecanismo COW (copy-on-write)

Ventajas:

- Reduce n.º operaciones de copia en memoria
 - Reduce n.º copias de marcos
 - Varios proc. comparten marco de lectura
- Si alguno intenta escribir \Rightarrow Kernel le asigna una copia

Gestión de pila

Rutina de falta de página $\xrightarrow{\text{recente}}$ SO gestiona la asig. de la pila
→ Idea: usuario intenta empujar dato en pila \Rightarrow sysv que no hay espacio

↓
La rutina de falta de página detecta intento en una dirección próxima a una req. de escritura con perm. de escritura y VM-GROWSDOWN

↓
Asigna página adyacente a páginas de pila

Demandas de página

Razones para no estar en mem.

- 1: Nunca ha sido accedida \Rightarrow su PTE se llena a 0
- 2: Se ha sacado a disco \Rightarrow No se llena a 0, indicador "Present" limpio

¿Cómo cargarla? Se convierte vm-area \rightarrow vm-obj \rightarrow page

- Nulo: apunta a función que carga la pag.
- Nulo: proc. anónima (El proceta ningún archivo)

→ Página cero \Rightarrow Dispara mecc. COW.

Reclamación de páginas

Mantiene un depósito mínimo de marcos libres

Se activa...

- Sincrona: falla una asignación de mem.
- Asíncrona: habrá Kswapd descubre que nº marcos por debajo del límite (marca baja de agua - page_low)

Se detiene...

Cuando se alcanza marca alta de agua (page_high)

Consideraciones para ajustar parám. \Rightarrow buen rendimiento

- Orden de examinación de los cachés
- dRU
- Determinar estado páginas (Sicas/no sicas)

Función: try_to_free_pages()

dRU divide las páginas de memoria en:

- lista activa págs recientemente accediidas
- lista inactiva págs no "

↳ De aquí sacar las págs a disco

Periodicamente hace un fijado entre estas

Cachés

- De páginas para acceso a archivos por ^{read(s)} write(s) _{mmayor(s)}
- De búferes: almacena metadatos de archivos ^{para rápida recuperación}
- De intercambio: evita errores entre proc. que intentan acceder a pág que se han sacado a disco

Una pág compartida puede estar para un proc. y sacada para otro