Diseño de sistemas operativos

Práctica 2
Memon

Fernando Pérez Costoya

Objetivo

- Monitor de fallos de página de un programa
  - ¿Cómo se comporta programa con *N* marcos y un algoritmo dado?

./memon_FIFO 8 ./prog arg1 arg2 ...

Fallos de página 41
Fallos no forzados 27
Fallos forzados 14
Fallos sin reemplazo 10
Fallos con reemplazo 31
Fallos sin lectura 3
Fallos con lectura archivo 25
Fallos con lectura swap 13
Escrituras en archivo 5
Escrituras en swap 12

Fundamento

- Monitor impide que programa acceda a sus regiones

  mprotect(dir, tam, permisos); (permisos=PROT_NONE)
- Cuando programa accede → SEGV
- Tratamiento de SEGV
  - Gestiona estadísticas
  - Debe comprobar que no es acceso erróneo del programa
    - Como sistemas reales: uso de tabla de regiones
  - Debe devolver permisos a página: los de la región
  - Si errores: *_exit*
  - Uso de *printf* facilita depuración

Versión preliminar

```
void fallo_pagina(void *dir_fallo) {
   printf("-------- fallo %p ----------\n", dir_fallo);
   Encontrar región y página a la que pertenece la página
   if (no pertenece a ninguna región) {
         printf("acceso a memoria inválido %p\n", dir_fallo); _exit(1);  }
   fallos_total++; ....
   printf("-------- dir mprotect %p ----------\n", dir);
   if (mprotect(dir_inicio_pagina, tam_pagina, permisos_region) <0) {
         perror("Error devolviendo permisos");  _exit(1);  }
   printf("-------- FIN fallo %p ----------\n", dir_fallo);
   return;
}
```

Estructuras de datos

- Tabla de regiones (TR) (**ya implementada**)
  - Vector de tamaño fijo
  - Cada entrada contiene la siguiente información:
    - usada, dir_inic, npags, comp./privada, permisos, fichero/anon.
    - Dirección de la tabla de páginas de esa región
- Tabla de páginas (TP) (parcialmente implementada)
  - Vector de tamaño dinámico (reservado en el *heap*)
  - Entrada contiene info de una página de la región (**A COMPLETAR**)
    - dir_inicial, puntero a la entrada de la t. regiones correspondiente
- Tabla de marcos (TM) (**ya implementada**)
  - Vector de tamaño fijo que almacena qué página reside en cada marco
  - No se debe acceder directamente sino usando funciones (marcos.h)
    - *reservar_marco_libre, reemplazo, rellenar_entrada_marco*, etc.

Eventos

- SEGV → rutina de fallo fallo_pagina (fallo.c) (**VACÍA**)
- Se ha creado una región (mapa.c):
  - Funcionalidad completa (Iniciar entrada en TR y reservar TP)
  - Por cada página llama a iniciar_entrada_tpag (**VACÍA**)
    - iniciar campos + impedir acceso a página
- Se ha eliminado una región (mapa.c):
  - Funcionalidad completa (liberar TP y entrada TR)
  - Por cada página llama a liberar_entrada_tpag (**VACÍA**)
    - Liberar marco ocupado por la página
- Ha cambiado tam. región cambio_tam_region (mapa.c): (**VACÍA**)
  - Si ha aumentado → similar a crear región
  - Si ha disminuido → similar a eliminar región

Funcionalidad pedida

1. Versión inicial: Aplicación directa de la idea básica
2. Versión intermedia: Control de la modificación de las páginas
3. Versión final: Algoritmo del reloj

Versión inicial

- Estadísticas: Fallos totales, con/sin reemplazo, forzados o no
- Rutina de fallo "similar" a la del SO:

```
void fallo_pagina (void *dir_fallo) {
        Comprobar que la dirección es válida
        Reservar un marco libre
        Si no hay ninguno {
            Aplicar el algoritmo de reemplazo que selecciona un marco
            Invalidar la página contenida en ese marco (mprotect) }
        Asociar la nueva página con el marco
        Poner como válida la página (mprotect)
}
```

Versión intermedia

- Estadísticas: lecturas/escrituras de m. secundaria
  - Lecturas: de fichero, de *swap* o sin lectura
  - Escrituras: en fichero o en *swap*
- Gestión de bit *mod* por cada página

```
void fallo_pagina (void *dir_fallo) {
    Si no está residente
        Código de la primera versión pero
            devolviendo permisos originales menos PROT_WRITE
    Si está residente
        Activar bit de modificado
        Devolver los permisos originales }
```


Versión final

- Algoritmo reloj (marcos.c): requiere gestionar bit *ref*
- Esbozo de la estrategia de gestión de *ref*
  - Al "traer" página: residente y referenciada
  - Alg. reloj desactiva *ref* + mprotect para impedir acceso
  - Fallo sobre página residente y no referenciada
    - Se activa *ref* + mprotect con permisos correspondientes