I OBJETIVOS

En esta práctica el alumno comprobará los beneficios de la memoria virtual para aumentar el grado de multiprogramación e implementará los mecanismos que se requieren para un sistema con memoria virtual utilizando los algoritmos de reemplazo de páginas estudiados.

II BIBLIOGRAFÍA

- William Stallings, "SISTEMAS OPERATIVOS",Prentice Hall, 4ª Ed.
- Silberschatz, Galvin, Gagne, "SISTEMAS OPERATIVOS", Limusa Wiley, 9ª Ed.

III RECURSOS

- Una estación de trabajo con Linux con su ambiente gráfico.
- Un editor de texto.
- Compilador de c.
- Los archivos que conforman el ambiente virtual `procesos.c`, `pagefault.c` `mmu.o` y `mmu.h` que están compactados en `mmu.zip`.

IV DESCRIPCIÓN DEL AMBIENTE

El ambiente virtual simula una computadora con una memoria física de 48 Kb. que se divide en 12 marcos de 4 Kb. En éste sistema se requiere que corran 4 procesos, cada uno de estos requiere un espacio de 24 Kb para almacenar datos. Este espacio de 24 Kb. que usa cada proceso, está dividido en 6 páginas de 4 Kb., que inicialmente no están asignadas a marcos de la memoria física y conforme los procesos las van requiriendo, se generan fallos de página de manera que el manejador de fallos de página les asigna marcos de la memoria física.

En este sistema de cómputo existe el problema de que no hay memoria suficiente para que los 4 procesos pueden ejecutarse simultáneamente, en el momento que un proceso requiere que el sistema operativo le asigne un marco de memoria física y no se le puede asignar, éste termina en un error.

A continuación se describen más detalladamente los bloques de construcción y el funcionamiento del ambiente virtual.

1 Bloques de construcción definidos por el ambiente

Cada proceso de este sistema tiene su tabla de páginas que es apuntada por `ptbr` (Process Table Begin Register), que es un apuntador a una estructura tipo `PROCESSPAGETABLE` cuya

definición está en el archivo `mmu.h` y se muestra en la Figura 1. El tamaño de esta tabla es de 6 elementos que también puede tomarse de la variable global `ptlr` (Process Table Lenght register).

```
struct PROCESSPAGETABLE {
    int presente;
    int modificado;
    int framenumber;
    unsigned long tarrived;
    unsigned long tlastaccess;

    int attached;   // No modificar
};
```

Figura 1. Estructura de la tabla de páginas de cada proceso.

El propósito de los campos que están definidos en la estructura `PROCESSPAGETABLE` son los que se muestran en la Tabla 1.
.

Campo	Propósito
`Presente`	1.- Si la página se encuentra asignada a un marco de la memoria física. 0.- Si la página no está asignada a un marco de la memoria física.
`modificado`	1.- Si la página ha sido modificada. 0.- Si la página no ha sido modificada.
`framenumber`	El número de marco asignado a la página. Si la página no tiene ningún marco asignado, este valor es -1.
`Tarrived`	Instante en nanosegundos en que la página fue asignada a un marco.
`tlastaccess`	Instante en nanosegundos del último acceso a la página.
`Attached`	Usado para control del mmu, no modificar.

Tabla 1. Descripción de los campos de la estructura de la tabla de páginas.

El sistema tiene una tabla `systemframetable` para administrar los marcos disponibles y asignados de la memoria principal, ésta tabla es un arreglo de estructuras del tipo `SYSTEMFRAMETABLE` cuya definición está en el archivo `mmu.h` y se muestra en la Figura 2. El tamaño de esta tabla es de 12 elementos que también puede tomarse de la variable `processpagetablesize`.

```
struct SYSTEMFRAMETABLE {
    int assigned;
    char *paddress; // No modificar
    int shmidframe;  // No modificar
```

```
};
```

Figura 2. Estructura de la tabla de marcos del sistema.

El propósito de los campos que están definidos en la estructura `SYSTEMFRAMETABLE` son los que se muestran en la Tabla 2.

Campo	Propósito
`assigned`	1.- Si el marco se encuentra asignado a una página de un proceso. 0.- Si el marco está libre.
`paddress`	Dirección de memoria del marco.
`shmidframe`	Usado para control del mmu, no modificar.

Tabla 2. Descripción de los campos de la estructura de la tabla de páginas.

2 Funcionamiento del ambiente virtual

2.1 Procesos y el acceso a la memoria

En el archivo `procesos.c` se encuentran definidas 4 funciones (`proc0()`, `proc1()`, `proc2()` y `proc3()`), cada una de ellas se refiere a los procesos que están en ejecución de manera concurrente. Los procesos en el ambiente virtual son identificados por un número entre 0 y 3, y este valor puede se encuentra la variable global `idproc` de cada proceso, esto es, el `idproc` del proceso 0 será igual a 0, el del proceso 1 será igual a 1, etc.

Los procesos tienen virtualmente disponible un espacio de 16 Kb. para almacenar datos, la dirección de inicio de este espacio temporal es apuntada por la variable global `base`, y las direcciones virtuales dentro de cada proceso son un desplazamiento a partir de la dirección contenida en `base`.

2.2 Rutina para el manejo de los fallos de página

La rutina para el manejo de fallos de página se encuentra definida en el archivo `pagefault.c`. Inicialmente las páginas de los procesos no se encuentran asignadas a un marco de página, tal que cuando un proceso intenta leer o escribir en una página no asignada ocurre un fallo de página haciendo que el control se transfiera a esta rutina.

Cuando un fallo de página ocurre, el control se transfiere a la rutina `pagefault()`. Esta rutina recibe en el parámetro `pag_del_proceso` la página que ocasionó el fallo y debe modificar la

tabla de páginas del proceso y la tabla de marcos del sistema de manera que a esta página se le asigne un marco de la memoria.

Para asignar un marco de la memoria principal a una página es necesario:

1. **Buscar un marco de la memoria disponible**. La tabla de marcos del sistema `systemframetable` es un arreglo donde el índice de cada elemento indica el número de marco. Es necesario recorrer el arreglo desde el primer elemento (0) hasta encontrar uno disponible, es decir, donde el campo `assigned` sea 0. Si no hay marcos disponibles es porque no hay memoria disponible en el sistema y la rutina debe regresar el error -1.

2. **Establecer el número de marco encontrado en la tabla de páginas**. La tabla de páginas del proceso `processpagetable` es un arreglo donde el índice es el número de página. Para asignarle un marco es necesario establecer en el campo `framenumber` el número de marco que fue encontrado en el paso anterior, y establecer el campo presente con el valor de 1.

Para sacar una página de la memoria principal es suficiente con poner el campo `presente` de la página en 0.

Si la rutina de manejo de fallos de página termina sin errores, debe regresar el entero 1.

V ACTIVIDADES

1 Preparación del ambiente

1.1 Compilación

El proceso de compilación requiere al menos dos pasos, se recomienda usar el `Makefile`[1] contenido en el conjunto de archivos para facilitar este proceso, aunque la descripción de cómo se realiza manualmente se describe a continuación:

1.- Compilar la rutina para el manejo de fallos de página. En este paso se generará un archivo .o que será encadenado al programa ejecutable.

[1] El `Makefile` es un archivo donde se describen los archivos que son generados a partir de otros, y el proceso para generarlos. Por ejemplo la compilación de un programa genera un archivo ejecutable a partir de uno o varios fuentes.

El `Makefile` es ejecutado por el programa `make` y además es capaz de comprobar la fecha y hora de actualización de los archivos para decidir cuales son los pasos que debe volver a realizar. Por ejemplo: volver a compilar si uno de los fuentes fue modificado después de la creación del ejecutable.

```
$ gcc –c pagefault.c
```

2.- Compilar el archivo donde están los procesos encadenándose a los archivos `pagefault.o` generado en el paso 1 y `mmu.o`.

```
$ gcc –o procesos procesos.c mmu.o pagefault.o
```

1.2 Ejecución

Este ambiente virtual tiene dos modos de ejecución, estos son el modo normal y el modo de depuración (debug). En modo normal solo se mostrará en pantalla los mensajes que envíen los procesos (`proc0()`, `proc1()`, `proc2()`, `proc3()`) con cualquiera de las llamadas al sistema para salida a pantalla (ejemplo: `printf()`). En modo de depuración, se mostrará en pantalla el detalle de lo ocurrido por los accesos a memoria realizados por los procesos.

Para ejecutarlo en modo normal solamente ejecute el programa `process`.

```
$ ./procesos
```

Para ejecutarlo en modo de depuración, ejecute el programa `process` añadiendo el parámetro `/debug`.

```
$ ./procesos /debug
```

2 Implementación de memoria virtual

Considere un sistema de memoria virtual donde **se utiliza un archivo en memoria secundaria como archivo de intercambio. Este archivo usualmente mide el doble de tamaño de la memoria principal** y se divide en secciones que corresponden al tamaño de un marco de página.

En este sistema de memoria virtual, todos los marcos de página de la memoria principal corresponden las primeras secciones del archivo de intercambio, y las secciones restantes del archivo de intercambio corresponden a los que llamamos marcos virtuales, como se muestra en la Figura 3.

Para esta implementación de memoria virtual donde hay marcos virtuales, el sistema provee la facilidad de que la tabla de marcos del sistema `systemframetable` tiene el espacio suficiente para administrar el doble de marcos de la memoria física.

Modifique el manejador de fallos de página de manera que utilice el archivo de intercambio para asignarles marcos virtuales a las páginas de los procesos que son

expulsadas y así mismo liberar marcos de la memoria principal para que puedan ser asignados a las páginas que los requieran. Implemente cualquiera de las políticas de reemplazo de páginas vista en clases, mida el número de fallos de página generados por los procesos.

Establezca un tamaño para el conjunto residente de cada proceso a un valor fijo, y cuando ocurra un fallo de página si el proceso tiene todos sus marcos asignados, reemplace una página del conjunto residente del proceso que produjo el fallo de página (asignación fija y alcance local).

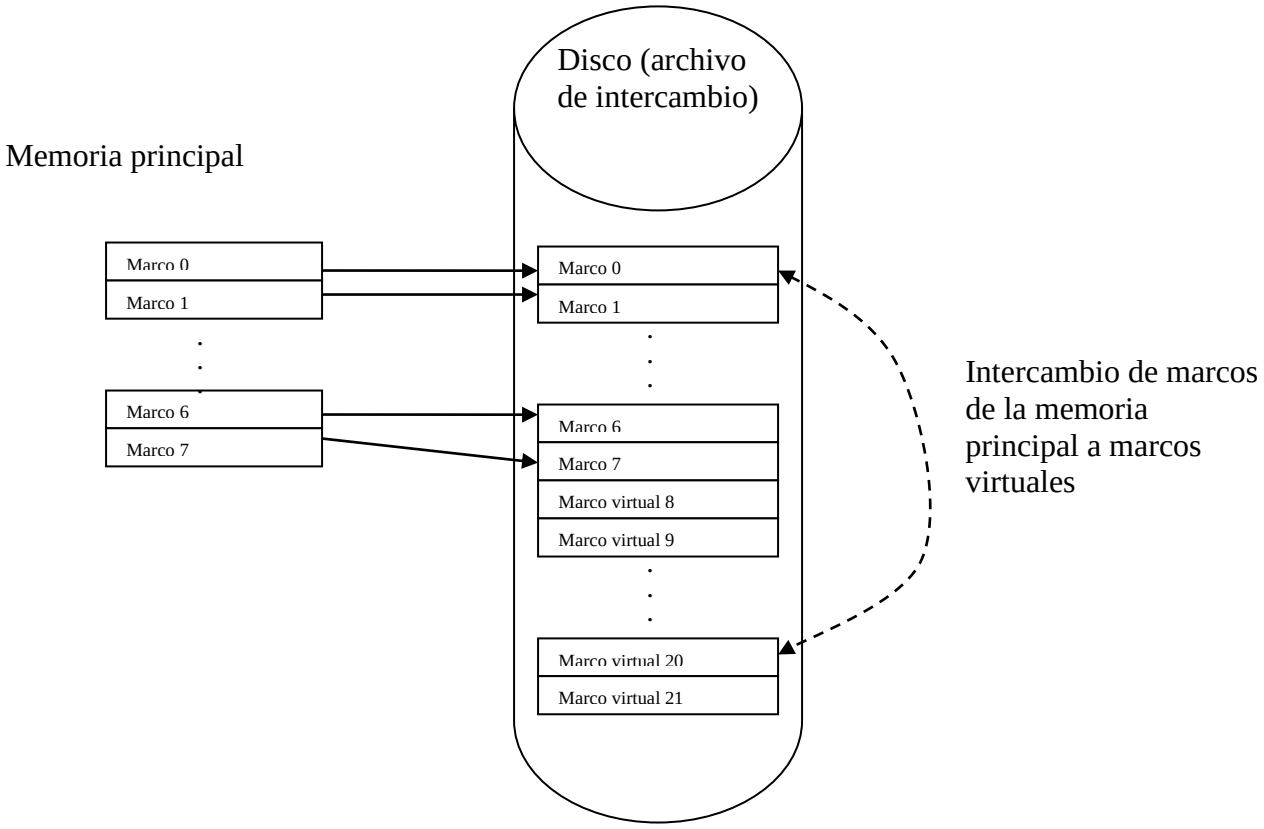

Figura 3. Mapeo de marcos al archivo `swap`

2.1 Funciones para realizar operaciones de lectura y escritura en el archivo `swap` para intercambio.

El ambiente virtual incluye algunas funciones para lectura y escritura de los marcos en el archivo de intercambio en memoria secundaria, estas funciones son las que se describen a continuación:

```
int copyframe(int sframe,int dframe);
```

La función `copyframe(int sframe,int dframe)` realiza la copia de un marco a otro marco dentro del archive de intercambio donde `sframe` es el marco fuente y `dframe` es el marco destino.

```
int writeblock(char *buffer, int dblock);
```

La función `writeblock(char *buffer, int dblock)` escribe un bloque que es del mismo tamaño de los marcos y páginas en el archivo de intercambio. El apuntador `buffer` apunta a la memoria lo que se tiene que escribir y `dblock` es el bloque a escrinir que corresponde a un marco en el archivo de intercambio.

```
int readblock(char *buffer, int sblock);
```

La función `readblock(char *buffer, int dblock)` lee un bloque que es del mismo tamaño de los marcos y páginas del archivo de intercambio. El apuntador `buffer` apunta a la memoria donde se almacenarán los datos leídos y `sblock` es el bloque a leer que corresponde a un marco en el archivo de intercambio.

```
int loadframe(int frame);
```

La función `loadframe(int frame)` transfiere al marco físico de la memoria física el contenido que corresponde al marco del archivo de intercambio. El parámetro `frame` es el marco físico que será cargado de su respaldo en el archivo de intercambio.

```
int saveframe(int frame);
```

La función `saveframe(int frame)` transfiere del marco `frame` de la memoria física al bloque del archivo de intercambio que le corresponde a ese mismo marco físico. El parámetro `frame` siempre debe ser un marco físico.

VI ENTREGA Y EVALUACIÓN

No incluya líneas de código en sus programas de las cuales desconozca su funcionamiento. El código no conocido será anulado en el funcionamiento de la práctica.

Aunque en semestres anteriores se han realizado prácticas similares a esta, hay aspectos que hacen que esta sea diferente. Cualquier evidencia que muestre el intento de entregar una práctica de semestre anterior será calificada como plagio.

1 Entrega y Revisión

Entregar en el apartado correspondiente de Moodle el archivo fuente `pagefaultVM.c` a más tardar el Domingo 18 de Noviembre a las 23:55 Hrs. La revisión se hará a partir de la semana 13

2 Equipos

Esta práctica se hará en equipos (máximo 2 integrantes), es necesario que en la revisión esté el equipo completo ya que el integrante que no se presente no tendrá calificación en la práctica (revisar el apartado de revisiones en evaluación).

Importante: Al indicarse que el trabajo debe ser desarrollado por equipos, se entiende que no se permite colaboración entre equipos, cualquier evidencia de esto será considerada plagio.

3 Evaluación

Puntualidad en las revisiones	El equipo estuvo completo y puntual en todas las sesiones de revisión.		Si hubo dos o más sesiones con el equipo, el equipo estuvo completo y puntual en casi todas las sesiones de revisión		Si solo hubo una sesión de revisión, el equipo no estuvo completo o no fue puntual. Si fueron dos o más sesiones de revisión, en más de una sesión el equipo no estuvo completo o fue puntual
	+5		+2.5		0
Especificaciones de entrega	La entrega del producto cumple con todas las especificaciones indicadas en el documento de la práctica, por ejemplo, los archivos se entregan de acuerdo a las formas indicadas en el documento de la práctica.				La entrega del producto no cumple con al menos una de las especificaciones indicadas en el documento de la práctica
	+5				0
Funcionamiento	El producto cumple con todas las especificaciones indicadas en el documento y no tiene fallas	El producto muestra una falla no esperada.	Se da una de las siguientes condiciones: -Está incompleto (falta máximo aprox 50%), pero lo demás puede funcionar bien. -Está completo pero muestra dos fallas	Se da una de las siguientes condiciones: -Está completo pero muestra 3 fallas o más. -Está incompleto (falta máximo aprox 50%) y además muestra fallas -Está incompleto (falta máximo aprox 66%)	El producto no funciona o no está incompleto (más del 66%).
	+80	+60	+40	+20	0
Interfaz con el usuario	El producto funciona y pudo ser utilizado sin necesidad de recibir indicaciones por el desarrollador, tiene instrucciones claras para ser utilizado.		El producto funciona, pero hubo necesidad de recibir alguna indicación para su uso por parte del desarrollador del producto		El producto carece de instrucciones claras para ser utilizado y requiere que alguno de los desarrolladores esté presente para su utilización o no puede utilizarse debido a que no está completo
	+5		+3.5		0
Claridad en el código	El código es claro, usa nombres de variables adecuadas, está debidamente comentado e indentado. Puede ser entendido por cualquier otra persona que no intervino en su desarrollo.		El código carece de claridad, puede ser entendido por cualquier persona ajena a su desarrollo pero con cierta dificultad.		El código carece de comentarios, está mal indentado, usa nombres de variables no adecuadas.
	+5		+3.5		0
Defensa del producto	Todos los que presentan la práctica son capaces de explicar cualquier parte del producto presentado		Uno de los que presenta la práctica muestra dudas sobre alguna parte del desarrollo del producto presentado		Más de un integrante no muestra evidencia de que conoce el producto, o si el trabajo fue individual, el desarrollador duda sobre el desarrollo del producto que presenta.
	x 1(puntos se multiplican por 1)		x 0.5 (puntos se multiplican por 0.5)		x 0 (puntos se multiplican por 0)
Sobresaliente 20 %	Tiene 1 en todos los puntos anteriores. El producto entregado es sobresaliente, muestra tener la calidad para ser expuesto como un producto representativo de la carrera Hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado				No tiene 1 en todos los puntos anteriores, o el producto entregado no es sobresaliente y no muestra tener la calidad para ser expuesto como un producto representativo de la carrera o no hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado
	+20				0