Sistemas Operativos

Proyecto: Paginación

Daniela Carolina Montenegro Pozo

18/09/2020

Índice

1. Antecedentes del problema

La paginación bajo demanda implica cargar una página solo cuando se necesita[1], el punto clave del proyecto es el manejo de los fallos.

Se tiene un archivo de direcciones virtuales y un archivo binario de backing store desde donde se extrae la data del proceso y se lo guarda en un frame, para que pueda ser accedido en algún momento de la ejecución. La primera vez que se llama a un proceso, al no existir una entrada con la página de su dirección virtual, se carga la información desde el backing store a un frame, una vez que se ha cargado en el frame, se puede disponer de esta información directamente desde la memoria principal, así que la próxima vez que la dirección virtual coincida en el número de página, no será necesario leer los datos de nuevo desde el backing store, sino que accede directamente con el número de frame y el offset. La mejor representación visual es la que se muestra en la imagen 1.

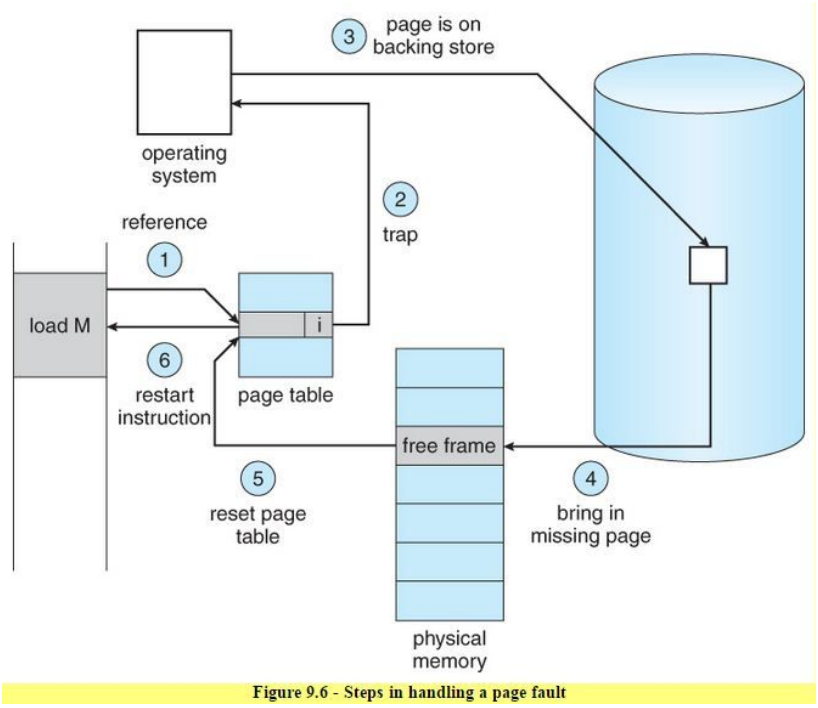

Imagen 1. Manejo de fallos de página[2]

2. Implementación y parámetros

Dadas las instrucciones del proyecto, se definieron las siguientes constantes globales:

- Tamaño en bits del offset

```
#define OFFSET 8
```

- Número de frames de memoria principal

```
#define FRAMES 256
```

- Tamaño del frame de memoria principal

```
#define FRAME_SIZE 256
```

- Tamaño de una página

```
#define PAGE_SIZE 256
```

Se definen también las siguientes variables globales:

- Contador para índice de frames de memoria

```
unsigned int indice_de_memoria = 0;
```

- Puntero al archivo de backing store

```
FILE * backing_store;
```

- Puntero al archivo de salida del programa

```
FILE * output;
```

Para la implementación se usó una lista de `<sys/queue.h>` para la tabla de páginas, el propósito era representar una búsqueda secuencial en dicha lista.

```
LIST_HEAD(list_of_pages, _Page) tabla_de_paginas;
```

La tabla de páginas usa structs de tipo Page, que contienen el número de página (vpn) y el índice del frame (pfn) donde se encuentra cargada la información.

```
typedef struct _Page {
```

```
    LIST_ENTRY(_Page) pptrs;

    unsigned int vpn;

    unsigned int pfn;

} Page;
```

Además se usó un arreglo multidimensional para representar la memoria principal, el propósito era hacer que el acceso al dato fuese directo, mediante los índices de número de frame y offset.

```
int memoria_principal[FRAMES][FRAME_SIZE];
```

Traducción de direcciones

Para la traducción se usaron dos operaciones:

- La operación right shift (>>), que es una operación bit por bit, que desplaza n bits hacia la derecha. Sirve para obtener el número de página[3], porque con este operando se desplazan 8 bits a la derecha. En este caso, la constante OFFSET es 8 (no debe ser confundida con el valor variable offset).

```
                    unsigned int vpn = address>>OFFSET
```

- El valor de la variable offset se obtiene a partir de la dirección al obtener el módulo entre la dirección y el tamaño de la página[3], por eso se usó el operador %.

```
                  unsigned int offset = address%PAGE_SIZE
```

Manejo de fallo de páginas

El manejo de fallos de página y entrega de datos de memoria principal ocurre en

```
bool insertOrCreatePage(unsigned int vpn, unsigned int offset)
```

La cual recibe el número de página y el offset como parámetros, retorna verdadero si la página se encontró, y false si no la encontró y tuvo que ser creada.

Para esta función se definieron las siguientes variables locales:

- Puntero a una página

```
    Page * p;
```

- Declaración del número entero que se solicitó

```
int requested;
```

- Dirección virtual completa

```
unsigned int vaddress = vpn*256+offset;
```

- Declaración de la dirección física

```
unsigned int paddress;
```

Primero se busca la página en la tabla de páginas, usando el puntero a una página y la tabla de páginas:

```
LIST_FOREACH(p, &tabla_de_paginas, pptrs)
```

Si el número de página de la página a la que apunta el puntero p (p->vpn) coincide con el número de página declarado en los parámetros, entonces se carga el número entero en requested a partir del número de frame (p->pfn) y el offset. La dirección física se guarda usando con el número de frame (p->pfn) y el offset.

```
if(p->vpn==vpn){
            //leer desde frame en memoria principal
            paddress = p->pfn*256+offset;
            requested = memoria_principal[p->pfn][offset];
```

Estos datos se guardan en el archivo output usando fprintf y se retorna true.

```
      fprintf(output,"Virtual address: %u Physical address: %u Value:
%d\n",vaddress,paddress,requested);
      return true;
      }
```

Si al terminar de recorrer la tabla de páginas no se encontró el número de página, entonces se carga la información desde el backing store a un frame usando `backingStoreData` que recibe como parámetros la el número de página y el índice del siguiente frame a ser usado.

```
backingStoreData(vpn,indice_de_memoria);
```

Luego se guarda crea la página, usando los mismos parámetros:

```
Page * pg = create_page(vpn,indice_de_memoria);
```

Y se obtiene el número con el offset y el índice de memoria:

```
requested = memoria_principal[indice_de_memoria][offset];
```

La dirección física corresponde al índice de memoria y al offset:

```
   paddress = indice_de_memoria*256+offset;
```

Se guarda la data en el archivo de salida con fprintf:

```
   fprintf(output,"Virtual address: %u Physical address: %u Value:
%d\n",vaddress,paddress,requested);
```

Se aumenta el contador de índice de memoria, porque ya se ocupó el frame:

```
indice_de_memoria++;
```

Se inserta la página a la tabla de páginas y se retorna con false, porque la página tuvo que ser creada.

```
LIST_INSERT_HEAD(&tabla_de_paginas, pg, pptrs);

return false;
```

Búsqueda en backing store y carga en frame

La función recibe como parámetros el número de página y el número de frame, no retorna nada.

```
void backingStoreData(unsigned int vpn, unsigned int pfn)
```

Primero se define la dirección base usando el número de página, en este caso lo llamé file_offset, porque corresponde al desplazamiento en el archivo.

```
   int file_offset = vpn * 256;
```

Declaro el arreglo de caracteres al cual se cargará los datos del backing store:

```
char num[FRAME_SIZE];
```

Apunto a la dirección exacta en el archivo del backing store con fseek, usando el file_offset:

```
fseek(backing_store, file_offset, SEEK_SET);
```

Leo y guardo en num, 256 caracteres de 1 byte desde el file_offset usando fread

```
fread(&num, 1, 256, backing_store);
```

Luego reseteo el puntero del archivo, para que vuelva al inicio:

```
rewind(backing_store);
```

Finalmente guardo los caracteres como números en el frame que corresponde al número de frame (pfn) que corresponde:

```
for (size_t i = 0; i < FRAME_SIZE; i++)

   {

       memoria_principal[pfn][i]=(int)num[i];

   }
```

Las funciones auxiliares se documentaron en el código, se omitirán porque no son parte del problema principal del proyecto.

3. Uso

Para compilar el programa use:

```
make pg
```

Para usar el programa use:

```
./pagingdemand [path_direcciones:String] [path_resultado:String]
```

Ejemplo:

```
./pagingdemand assets/addresses.txt data.txt
```

4. Limitaciones

En un inicio la mayor dificultad fue entender cómo leer el archivo binario, la forma en la que lo hice por primera vez fue por la terminal con el comando

```
xxd -b assets/BACKING_STORE.bin
```

De esa forma pude visualizar los números en binario y tener una guía inicial.

Para leer el archivo binario se necesitaba aumentar la b en el segundo parámetro de fopen:

```
backing_store = fopen ("./assets/BACKING_STORE.bin", "rb+");
```

Luego el problema fue transformar los bytes a números, porque debían ser leídos uno por uno, como caracteres, entonces primero se lee guardan los bytes en un arreglo de caracteres y después se usa casting de char a int. Eso se refleja en la parte 2 “Implementación y parámetros”, en la sección ”Búsqueda en backing store y carga en frame”. Para no tener que hacer casting de nuevo, el arreglo multidimensional que se usa como memoria principal es un arreglo multidimensional de enteros, y no de caracteres.

La traducción de direcciones no fue compleja, gracias a la referencia [3] y a que conocía la existencia de los operadores bitwise de desplazamiento, dado que los use bastante en lenguaje ensamblador, para la clase de Organización de Computadores.

Es muy probable que el programa no funcione adecuadamente si no se provee un archivo de direcciones que no contenga el formato adecuado, o un backing store distinto, procure que sea un archivo con una lista de enteros y un backing store con caracteres en binario. Guarde siempre el backing store como BACKING_STORE.bin en el directorio assets.

5. Pruebas y Evidencia de ejecución

El programa fue desarrollado en una máquina Xenial Xerus (Ubuntu 16.04) nativo y en una máquina virtual Focal Fossa (Ubuntu 20.04). A continuación se muestran los casos de prueba con su respectiva evidencia de ejecución en la máquina virtual Focal Fossa (Ubuntu 20.04).

Compilación sin warnings con gcc 7.x o superior

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ gcc --version
gcc (Ubuntu 9.3.0-10ubuntu2) 9.3.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ uname -a
Linux dcmontenegro-VirtualBox 5.4.0-47-generic #51-Ubuntu SMP Fri Sep 4 19:50:52 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$
```

Compilación con make

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ make pg
Compilando...
gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$
```

Ejecución exitosa

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ make pg
Compilando...
gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ./pagingdemand assets/addresses.txt data.txt
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  data.txt  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ tail data.txt --lines=10
Virtual address: 48065 Physical address: 25793 Value: 0
Virtual address: 6957 Physical address: 26413 Value: 0
Virtual address: 2301 Physical address: 35325 Value: 0
Virtual address: 7736 Physical address: 57912 Value: 0
Virtual address: 31260 Physical address: 23324 Value: 0
Virtual address: 17071 Physical address: 175 Value: -85
Virtual address: 8940 Physical address: 46572 Value: 0
Virtual address: 9929 Physical address: 44745 Value: 0
Virtual address: 45563 Physical address: 46075 Value: 126
Virtual address: 12107 Physical address: 2635 Value: -46
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$
```

Omitir archivo de direcciones

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ make pg
Compilando...
gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ./pagingdemand

[ERROR] Archivo de direcciones no definido.

 ********************************************************************************

 Proyecto del Segundo Parcial
 Paginación y Memoria Virtual
 Autor: Daniela Montenegro
 Sistemas Operativos 2020-1S

 Uso:

 ./pagingdemand ./assets/addresses.txt data.txt

 ./pagingdemand [path_direcciones:String] [path_resultado:String]

 ********************************************************************************
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$
```

Omitir nombre de archivo de salida

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ make pg
Compilando...
gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ./pagingdemand assets/addresses.txt

[ERROR] Archivo destino no definido.

 ********************************************************************************

 Proyecto del Segundo Parcial
 Paginación y Memoria Virtual
 Autor: Daniela Montenegro
 Sistemas Operativos 2020-1S

 Uso:

 ./pagingdemand ./assets/addresses.txt data.txt

 ./pagingdemand [path_direcciones:String] [path_resultado:String]

 ********************************************************************************
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  Makefile  pagingdemand  README.md  src
```

Archivo de direcciones no encontrado

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ make pg
Compilando...
gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ./pagingdemand addresses.txt data.txt

[ERROR] Archivo de direcciones ilegible.

 ********************************************************************************

 Proyecto del Segundo Parcial
 Paginación y Memoria Virtual
 Autor: Daniela Montenegro
 Sistemas Operativos 2020-1S

 Uso:

 ./pagingdemand ./assets/addresses.txt data.txt

 ./pagingdemand [path_direcciones:String] [path_resultado:String]

 ********************************************************************************
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  Makefile  pagingdemand  README.md  src
```

Backing store no encontrado

Nota: El backing store siempre debe estar en la carpeta assets para que el programa se ejecute correctamente, para esta prueba solo moví el backing store desde assets a la carpeta raíz.

```
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  BACKING_STORE.bin  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ make pg
Compilando...
gcc -std=c11 -o pagingdemand src/pagingdemand.c
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ./pagingdemand assets/addresses.txt data.txt

[ERROR] Backing store no encontrado.

Verifique la existencia del archivo BACKING_STORE.bin en la carpeta assets

 ********************************************************************************

 Proyecto del Segundo Parcial
 Paginación y Memoria Virtual
 Autor: Daniela Montenegro
 Sistemas Operativos 2020-1S

 Uso:

 ./pagingdemand ./assets/addresses.txt data.txt

 ./pagingdemand [path_direcciones:String] [path_resultado:String]

 ********************************************************************************
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$ ls
assets  BACKING_STORE.bin  Makefile  pagingdemand  README.md  src
dcmontenegro@dcmontenegro-VirtualBox:~/Escritorio/repos/OS20201PFVM$
```

Bibliografía

[1]M. Fienup, "Virtual Memory Overheads", *Cs.uni.edu*, 2020. [Online]. Available: http://www.cs.uni.edu/~fienup/cs142f03/lectures/lec16_OS_virtual_memory.htm . [Accessed: 17- Sep- 2020].

[2]"Implementation of Demand Paging in OS » PREP INSTA", PREP INSTA, 2020. [Online]. Available: https://prepinsta.com/operating-systems/implementation-of-demand-paging/ . [Accessed: 17- Sep- 2020].

[3]"Operating Systems", Www2.cs.uregina.ca , 2020. [Online]. Available: http://www2.cs.uregina.ca/~hamilton/courses/330/notes/memory/paging.html. [Accessed: 31- Aug- 2020].