UASLP
Universidad Autónoma
de San Luis Potosí

FACULTAD DE
INGENIERÍA

Práctica 3 . Paginación por demanda pura con algoritmo de reemplazo FIFO.

Sistemas operativos B.
Profesora: M.I. Ortiz Hernández Marcela.
grupo: 2402-03.
Semestre: 2019-2020/II.

Cantu Olivares Pedro de Jesus..

21/Mayo/2020.

1.- Resumen:

Objetivo:
Poder ejecutar el programa halt, matmult y sort en nachos implementando memoria virtual con paginación por demanda pura utilizando el algoritmo de reemplazo FIFO.

Características algoritmo FIFO (First In First Out):

- Algoritmo sencillo de reemplazo.
- Se reemplaza la página que haya llegado primero (la primer página en ser cargada en memoria es la página víctima).
- Presenta Anomalía de Belady.

2.- Instrucciones:

1. Tomando como base la práctica 2 implementar el algoritmo de reemplazo FIFO, siguiendo el procedimiento Pasos para el reemplazo de páginas
a) Atender la excepción de fallo de página y encontrar la ubicación de la página deseada en disco (archivo de intercambio)
b) Encontrar un marco libre: • Si hay un marco libre, usarlo • Si no hay un marco libre, usar el algoritmo de reemplazo FIFO para elegir la página víctima • Escribir la página víctima en disco(solo si fue modificada), actualizar tabla de páginas y de marcos c) Traer la página deseada al (nuevo) marco libre; actualizar las tablas de páginas y de marcos d) Continuar con la ejecución del proceso

2. Cuando se haya elegido una víctima con el algoritmo de reemplazo se deberá verificar si esta página fue modificada para respaldarla en su archivo de intercambio(swap out), en caso contrario, no se hace nada debido a que la información que está en memoria es idéntica a la del archivo de intercambio.

3. Se debe identificar la variable del SO que tiene la cuenta de las lecturas a disco duro e incrementarla cada que se realice un intercambio hacia adentro

4. Se debe imprimir en el resumen final de la máquina el número total de fallos de páginas, el número de lecturas y escrituras en disco al finalizar la ejecución del programa(como se muestra), donde, a) el número de lecturas tiene que ser igual al número de fallos y, b) el número de escrituras tiene que

ser menor al número de fallos, ya que solamente se deben escribir en disco aquellas páginas que hayan sido modificadas. matmult y sort deberán ejecutarse y finalizar correctamente, mostrando el resumen de la máquina como se muestra en el punto anterior.

5. La salida que tendrá que imprimirse en terminal es el vpn de cada página que causó el fallo de página separadas por comas.

3.- Liste todos los archivos que se modificaron para esta práctica y una breve descripción de modificación:

1.- archivo modificado: ./nachos/userprog/stats.h:
Descripción:
Definición de la variable para contar el marco a asignar.
Código:

```
  /*********************************
    Practica 3
    *********************************/
    int contadorMarco;
```

2.-archivo modificado: ./nachos/userprog/stats.cc:
Descripción:
Se agrego una variable para controlar el número de marco se se le asigna a una página que entra con fallo.
Código:

```
Statistics::Statistics()
{
    totalTicks = idleTicks = systemTicks = userTicks = 0;
    numDiskReads = numDiskWrites = 0;
    numConsoleCharsRead = numConsoleCharsWritten = 0;
    numPageFaults = numPacketsSent = numPacketsRecvd = 0;
    /****************************
    Practica 3
    ***********************/
    contadorMarco = 0;
}
```

3.-archivo modificado: ./nachos/userprog/addSpace.h:
Descripción:
Declaración del método para hacer swapOut.
Código:

```
/*****************************+
    Practica 3:
    *****************************/
    bool swapOut();
```

4.-archivo modificado: ./nachos/userprog/addSpace.cc:
Descripción:
Definicion del metodo para hacer swapOut, para a clase AddrSpace.
Código:

```
bool
AddrSpace::swapOut()
{
    int indicePagina = -1;

    for(int i = 0 ; i < numPages ; i++)
    {
        if(pageTable[i].physicalPage ==
stats->contadorMarco && pageTable[i].valid == TRUE)
        {
            indicePagina = i ;
            break;
        }
    }

    if(indicePagina <= -1)
    {
        printf("Pagina no encontrada\n");
    }
    else
    {
        printf("Haciendo swapOut a
%s\n",machine->swapFileName );
        if(pageTable[indicePagina].dirty)//la pagina esta
sucia.
        {
```

```
            OpenFile *swp =
fileSystem->Open(machine->swapFileName);   //abrimos el
archivo
            int direccionBaseDeMarco =
stats->contadorMarco * PageSize;
            if(swp == NULL)
            {
                printf("\nswabFile no abrio\n");
                return false;
            }
            else
            {
                printf("Escribiendo en la direccion %d de
la memoria principal\nTamaño de escritura: %d\nDesde la
direccion %d del archivo de
intercambio.\n",direccionBaseDeMarco,PageSize,indicePagin
a* PageSize);

swp->WriteAt(&(machine->mainMemory[direccionBaseDeMarco])
, PageSize, indicePagina* PageSize);
                stats->numDiskWrites++;
            }
            delete swp; //cerramos el archivo

        }

        pageTable[indicePagina].valid = FALSE;
        pageTable[indicePagina].dirty = FALSE;

        return true;
    }
    return false;
}
```

4. Tablas de resultados:

Halt/Marcos	Fallos	Lecturas	Escrituras
64	3	3	0
32	3	3	0
16	3	3	0
8	3	3	0
4	3	3	0

Matmult/Marcos	Fallos	Lecturas	Escrituras
64	47	47	0
32	110	110	48
16	8,960	8,960	1,090
8	23,375	23,375	5,208
4	82,981	82,981	17,611

Sort/Marcos	Fallos	Lecturas	Escrituras
64	40	40	0
32	3,309	3,309	2,879
16	10,954	10,954	8,367
8	19,467	19,467	10,821
4	1,589,336	1,589,336	540,722

5. Conclusiones

El algoritmo de remplazo FIFO es muy simple pero se puede complicar a medida que se reduce el número de marcos de la arquitectura.