UASLP
Universidad Autónoma
de San Luis Potosí

FACULTAD DE
INGENIERÍA

Práctica 2.Paginación por demanda pura.

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

Cantu Olivares Pedro de Jesus.

21/Mayo/2020.

Documentación de la práctica 2.

1. Descripción y características de:

• Paginación:

Permite que la memoria de un proceso no sea contigua y que a un proceso se le asigne memoria física donde esté disponible. Cada proceso tiene su propia tabla de páginas, cuando se carga todas sus páginas en la memoria principal, se crea; cada tabla contiene el número de marcos de página.

• Paginación por demanda:

Disminuye los tiempos de respuesta y aumenta la cantidad de programas en memoria.
Se intercambian páginas que no son necesarias.

• Paginación por demanda pura:

Paginación por demanda pura: Para indicar que está en memoria. Consiste en que el SO divide dinámicamente los programas en unidades fijas las cuales va a manipular la RAM del equipo.

2. Descripción de lo que es el Archivo de intercambio y cual es su funcionalidad para esta práctica.

Es un archivo que se almacena en el disco duro en donde se ponen datos de la memoria RAM y sirve para simular más RAM de la que se tiene. En esta práctica fue un estímulo de memoria para poder realizar lo que se denomina como swap in en el cual se guarda externamente y se vuelve a tomar para su próxima ejecución con swap out, para así poder hacer que el programa se ejecutará como corresponde.

3. Descripción del intercambio hacia dentro (swap in).

Consiste en traer de vuelta un proceso habiendo realizado el swap out antes y traerlo a la memoria principal para continuar su ejecución.

4. Descripción del intercambio hacia fuera (swap out).

Proceso puede ser cambiado temporalmente fuera de la memoria principal a un disco de almacenamiento.

5. Descripción de la práctica.

La práctica consistió en sí en implementar el denominado método de paginación por demanda pura, en el cual debía de mostrar los diferentes fallos que pudieran causarse dentro del mismo procedimiento en sí y dando una impresión de los fallos que se encuentren, acto seguido debe de verificar cada uno de los detalles de dichos errores y cada uno de sus componentes, mientras ello se deberá de manejar diferentes tamaños de marcos para verificar sus diferentes salidas de información y comparar cada uno de ellos. En el código se deberá de manejar distintas modificaciones en los cuales se deberá de

implementar el fallo de página antes mencionado para su contador e impresión de datos. Implementación de paginación por demanda pura SIN algoritmo de reemplazo en nachos.

6. Escriba la cantidad total de horas que trabajaron(efectivas de trabajo) en esta práctica.
La cantidad total fue de entre 12-15 hrs.

7. Describa a detalle como fue la organización y comunicación de los integrantes para el desarrollode la práctica.

Bueno principalmente nosotros nos mantuvimos en comunicación mediante un grupo de redes sociales en el cual nos mencionamos los distintos detalles que podría llevar la práctica, como poder resolverla, manejamos dudas y resoluciones en ello, nos definimos las distintas actividades en las cuales podíamos ayudar, verificamos el código y vimos algunas partes en las cuales nos podíamos ayudar.

9. Realice una tabla donde describa a detalle para cada integrante del equipo la(s) actividad(es) que
desarrolló en esta práctica y el día en que fue realizada.

Pedro	Encargado principal de organizar al equipo, fue el que definió los métodos en translate.cc, y addrspace.cc, además de ayudar con los diferentes archivos, también dio varias ideas de como poder resolver y dar coherencia en dichos archivos.
Connie	Encargada del archivo exception.cc y translate.cc además de aportar ideas de la realización de dichos métodos dentro de los demás archivos
Manuel	Encargado del archivo exception.cc y ayuda en el archivo addrspace.cc junto con addrspace.h además de definir diferentes ideas de la realización de la práctica

10. Dibuje la tabla de páginas para cada archivo de prueba indicando el contenido final después de ejecutar la práctica y trabajando con 16 marcos. Especifique cada uno de los campos de la tabla de páginas.

1.- Halt:

Índice de página.	No. Marco.	Validez
0	0	1
1	1	1
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	2	1

2.-Matmult:

Índice de página.	No. Marco.	Validez
0	0	1
1	1	1
2	3	1
3	5	1
4	8	1
5	0	0
6	0	0
7	0	0
8	4	1
9	10	1

10	13	1
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	6	1
21	9	1
22	12	1
23	15	1
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0
31	0	0
32	0	0
33	7	1
34	11	1
35	14	1
36	0	0
37	0	0

38	0	0
39	0	0
40	0	0
41	0	0
42	0	0
43	0	0
44	0	0
45	0	0
46	0	0
47	0	0
48	0	0
49	0	0
50	0	0
51	0	0
52	0	0
53	2	1

3.- Sort:

Índice de página.	No. Marco.	Validez
0	0	1
1	1	1
2	3	1
3	0	0
4	0	0
5	0	0
6	4	1
7	5	1
8	6	1

9	7	1
10	8	1
11	9	1
12	10	1
13	11	1
14	12	1
15	13	1
16	14	1
17	15	1
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	0	0
28	0	0
29	0	0
30	0	0
31	0	0
32	0	0
33	0	0
34	0	0
35	0	0
36	0	0

37	0	0
38	0	0
39	0	0
40	0	0
41	0	0
42	0	0
43	0	0
44	0	0
45	0	0
46	2	1

11. Para cada modificación realizada para esta práctica realice lo siguiente

• Describa a detalle la modificación realizada.
• Especifique el nombre de los archivos que se modificaron.
• Indique los métodos y/o funciones realizadas o modificadas.
• Anexe el código.

1.- Archivo modificado: ./nachos/machine/translate.cc

Descripción:

Se modificó el método translate de Machine para que imprima lo que va pasando en los fallos.

Código:

```
ExceptionType
Machine::Translate(int virtAddr, int* physAddr, int size, bool
writing)
{
    int i;
    unsigned int vpn, offset;
    TranslationEntry *entry;
    unsigned int pageFrame;

     DEBUG('a', "\tTranslate 0x%x, %s: ", virtAddr, writing ?
"write" : "read");

// check for alignment errors
     if (((size == 4) && (virtAddr & 0x3)) || ((size == 2) &&
(virtAddr & 0x1))){
```

```
     DEBUG('a', "alignment problem at %d, size %d!\n",
virtAddr, size);
     printf("\nException : AddressErrorException\n");
     return AddressErrorException;
    }

      // we must have either a TLB or a page table, but not
both!
    ASSERT(tlb == NULL || pageTable == NULL);
    ASSERT(tlb != NULL || pageTable != NULL);

     // calculate the virtual page number, and offset within
the page,
     // from the virtual address
    vpn = (unsigned) virtAddr / PageSize;
    offset = (unsigned) virtAddr % PageSize;

    /**********************************
    Practica 2 Impresion de informacion.
    **********************************/
//    printf("\n::TRANS::\n");
    /*printf("Direccion Virtual: %d\n", virtAddr);
    printf("Tamaño de pagina: %d\n",PageSize);
    printf("vpn calculado: %d\n",vpn);
    printf("offset calculado: %d\n",offset);
    */

    if (tlb == NULL)
    {          // => page table => vpn is index into table
          if (vpn >= pageTableSize)
          {
               DEBUG('a', "virtual page # %d too large for page
table size %d!\n",virtAddr, pageTableSize);
              printf("\nException : AddressErrorException\n");
              return AddressErrorException;
          }
          else if (!pageTable[vpn].valid)
          {
               printf("\n::TRANS_FAIL::\n");
               printf("Fallo   #     %d\n",stats->numPageFaults
+1);
```

```
                printf("Direccion    Virtual(Logica):    %d\n",
virtAddr);
                printf("offset(Desplazamiento)       calculado:
%d\n",offset);
          printf("Tamaño de pagina: %d\n",PageSize);
                printf("vpn calculado: %d\n",vpn);

                /*****************************
                Practica 2. Fallo de pagina.
                *****************************/
                if(stats->numPageFaults < NumPhysPages)
                {
                     if(currentThread->space->swapIn(vpn))
                     {
                          //asignar el marco a la pagina.

                          pageTable[vpn].physicalPage         =
stats->numPageFaults;
                          printf("Nuevo marco para la pagina %d
: %d\n",vpn,pageTable[vpn].physicalPage);
                          //Hacer la pagina valida.
                          pageTable[vpn].valid = TRUE;
                     }
                     else
                     {
                          printf("::::ERROR::::\nNo   se   pudo
hacer  swapIn:  -vpn:  %d  -numPageFaults:  %d  -NumPhysPages:
%d",vpn,stats->numPageFaults,NumPhysPages );
                     }

                }
                else
                {
                     printf("::::ERROR::::\nNo  hay  suficientes
marcos para hacer otro swapIn\n");
                     //printf("\n\n   Imprimiendo   tabla   de
paginas final...\n\n");
                     //printf("\n\nTabla de paginas:\n");
                     /*if(stats->numPageFaults >= 1)
                printf("Indice \tNo.Marco\tBit Validez\n");
                if(stats->numPageFaults >= 1)
```

```
                    for (i = 0; i < stats->numPaginasEntabla;
i++)
                {

                                        printf("%d \t %d \t\t
%d\n",pageTable[i].virtualPage,pageTable[i].physicalPage,pageT
able[i].valid);//imprimir la informacion de la pagina actual
con el indice i.
                }*/
               }
               //imprimir y sumar uno al numero de fallosa
                    printf("Fallo      #      %d      Fin.\n",
++stats->numPageFaults);

                    /*if(stats->numPageFaults >= 2)
              printf("Indice \tNo.Marco\tBit Validez\n");
              if(stats->numPageFaults >= 2)
                    for (i = 0; i < stats->numPaginasEntabla;
i++)
                {

                                        printf("%d \t %d \t\t
%d\n",pageTable[i].virtualPage,pageTable[i].physicalPage,pageT
able[i].valid);//imprimir la informacion de la pagina actual
con el indice i.
                }
                */

              DEBUG('a', "virtual page # %d too large for page
table size %d!\n",virtAddr, pageTableSize);
              return PageFaultException;
          }

          entry = &pageTable[vpn];
          printf("\n::TRANS::\n");
          printf("Direccion Virtual(Logica): %d\n", virtAddr);
          printf("Numero de pagina:%d\n",entry->physicalPage);
          printf("offset(Desplazamiento)              calculado:
%d\n",offset);
     printf("Tamaño de pagina: %d\n",PageSize);
          printf("vpn calculado: %d\n",vpn);
```

```
          printf("Direccion                                    Fisica:
%d\n",(entry->physicalPage* PageSize + offset) );


          printf("\n");
    }
    else
    {
              for (entry = NULL, i = 0; i < TLBSize; i++)
         if (tlb[i].valid && (tlb[i].virtualPage == vpn))
         {
               entry = &tlb[i];              // FOUND!
               break;
          }
          if (entry == NULL)
          {                    // not found
                DEBUG('a', "*** no valid TLB entry found for
this virtual page!\n");

               return PageFaultException;          //    really,
this is a TLB fault,
                                    //  the  page  may  be  in
memory,
                                    // but not in the TLB
          }
    }

    if (entry->readOnly && writing)
    {     // trying to write to a read-only page
     DEBUG('a',  "%d  mapped  read-only  at  %d  in  TLB!\n",
virtAddr, i);

     return ReadOnlyException;
    }
    pageFrame = entry->physicalPage;

     /************************************************
     Practica0.
     ************************************************/
     //impresion de direcciones logicas
```

```
     //printf("%d     \t\t\t     %d     \t\t\t     %d     \t\t\t
%d\n",virtAddr,pageFrame,offset,(pageFrame    *    PageSize    +
offset));

     //printf("Dirección    lógica    \t    No.Pagina(p)    \t
Desplazamiento(d)                \t                Dirección
Fisica\t\n",virtAddr,virtAddr,pageFrame,offset,pageFrame      *
PageSize + offset);



     // if the pageFrame is too big, there is something really
wrong!
     // An invalid translation was loaded into the page table
or TLB.
    if (pageFrame >= NumPhysPages) {
     DEBUG('a',   "***   frame   %d   >   %d!\n",   pageFrame,
NumPhysPages);
     return BusErrorException;
    }
    entry->use = TRUE;        // set the use, dirty bits
    if (writing)
     entry->dirty = TRUE;
    *physAddr = pageFrame * PageSize + offset;
        ASSERT((*physAddr  >=  0)  &&  ((*physAddr  +  size)  <=
MemorySize));
    DEBUG('a', "phys addr = 0x%x\n", *physAddr);
    return NoException;
}
```

2.- Archivo modificado: ./nachos/machine/addrSpace.h
Descripción:
Metodo en la clase AddrSpace para hacer el swapIn dado un vpn. Se llama desde translate.cc.
Código:

```
class AddrSpace {
  public:
    AddrSpace(OpenFile *executable,char* filename);    //
Create an address space,
                        // initializing it with the program
```

```
                        // stored in the file "executable"
    ~AddrSpace();             // De-allocate an address space

    void InitRegisters();          //   Initialize   user-level
CPU registers,
                        // before jumping to user code

    void SaveState();              //   Save/restore   address
space-specific
    void RestoreState();      // info on a context switch
    /*******************************************************
    Practica 2:
    *******************************************************/
    bool swapIn(int vpn);//funcion para el swap In

  private:
    int fallosDePagina = 0;
    TranslationEntry *pageTable;   // Assume linear page table
translation
                        // for now!
    unsigned int numPages;         // Number of pages in the
virtual
                        // address space
};
```

3.- Archivo modificado: ./nachos/machine/addrSpace.h
Descripción:
Declaración de variables para el archivo de intercambio.
Código:

```
/************************************+
Practica 2
*************************************/
OpenFile *swapOpenFile;
```

4.- Archivo modificado: ./nachos/machine/addrSpace.h
Descripción:
Cambio en el constructor de Addrspace para inicializar la tabla de páginas.
Código:

```
AddrSpace::AddrSpace(OpenFile *executable, char* filename)
{

    /**************************************************
    Practica 1: para crear el archivo de intercambio
    **************************************************/
    char swapPath [strlen(filename)+4] = "";
    strcat(swapPath,filename);
    strcat(swapPath,".swp");
    //printf("%s\n",swapPath);

    if(!fileSystem->Create(swapPath,executable->Length()-40))
    {
          printf("\nNo se pudo crear el archivo de intercambio
%s\n", swapPath );
    }
    else
    {
        machine->swapFileName = new char[strlen(swapPath)];
        strcpy(machine->swapFileName,swapPath);
         //printf("\n%s |-| Tam original::::: %d \n",swapPath,
strlen(swapPath));
                    //printf("\n%s   |-|   Tam   otro:::::   %d
\n",machine->swapFileName, strlen(machine->swapFileName));
                              /*OpenFile    **/swapOpenFile    =
fileSystem->Open(swapPath);

        if(swapOpenFile == NULL)
        {
            printf("\nEl archivo de intercambio no existe\n");
        }
        else
        {
            char *aux;
            aux = new char[executable->Length()-40];

                                              int    auxInt1    =
executable->ReadAt(aux,executable->Length()-40,40);
```

```
            if(auxInt1 > 0)
            {
                                                int   auxInt2   =
swapOpenFile->Write(aux,executable->Length()-40);
                delete swapOpenFile;
                if(auxInt2 <= 0)
                {
                          printf("\nNo se pudo escribir en el
archivo de intercambio\n");
                }
            }
            else
            {
                      printf("\nNo se pudo hacer lectura del
ejecutable\n");
            }
        }
    }


    NoffHeader noffH;
    unsigned int i, size;

    executable->ReadAt((char *)&noffH, sizeof(noffH), 0);
    if ((noffH.noffMagic != NOFFMAGIC) &&
          (WordToHost(noffH.noffMagic) == NOFFMAGIC))
     SwapHeader(&noffH);
    ASSERT(noffH.noffMagic == NOFFMAGIC);

// how big is address space?
          size   =   noffH.code.size   +   noffH.initData.size   +
noffH.uninitData.size
               + UserStackSize;    // we need to increase the
size
                              // to leave room for the stack

    numPages = divRoundUp(size, PageSize);

    stats->numPaginasEntabla = numPages;

    /************************************************
```

```
    Practica 0 :
    ***********************************************/
    printf("\nTamaño del proceso: %d Bytes.\n",size);//imprime
el tamaño del proceso.
          printf("\nNumero de paginas para el proceso:
%d.",numPages);//imprime el numero de paginas necesarias para
cargar el proceso.

    size = numPages * PageSize;

    /*********************************
    Practica 2
    *********************************/
/*
if(numPages > NumPhysPages) // Para evitar que se impriman las
tablas en los procesos que usen mas paginas que las que se
tienen.
{
      printf("\n\nEl tamaño del proceso excede las paginas
disponibles( %d ), y no se ha implementado memoria virtual por
lo que no puede ser cargado.\n\n",NumPhysPages);
}
*/
    //ASSERT(numPages <= NumPhysPages);      //   check   we're
not trying
                              // to run anything too big --
                              // at least until we have
                              // virtual memory

    DEBUG('a', "Initializing address space, num pages %d, size
%d\n",numPages, size);

    //Practica 0. para imprimir la tabla de paginas.
    printf("\n\nTabla de paginas:\n");
    printf("Indice \tNo.Marco\tBit Validez\n");

    // first, set up the translation
    pageTable = new TranslationEntry[numPages];
    //pageTable = new TranslationEntry[NumPhysPages];
    /*********************************************
    Practica 2
    *********************************************/
```

```
    for (i = 0; i < numPages; i++)
    {
     pageTable[i].virtualPage = i; // for now, virtual page # = phys page #
     //pageTable[i].physicalPage = i;
     pageTable[i].valid = FALSE;
     pageTable[i].use = FALSE;
     pageTable[i].dirty = FALSE;
     pageTable[i].readOnly = FALSE;   // if the code segment was entirely on
                          // a separate page, we could set its
                          // pages to be read-only
    //Practica0.
                           printf("%d      \t      %d      \t\t %d\n",pageTable[i].virtualPage,pageTable[i].physicalPage,pageTable[i].valid);//imprimir la informacion de la pagina actual con el indice i.
    }

    //Practica0
    //printf("\nMapeo de direcciones logicas:\n");
          //printf("Dirección  lógica  \t  No.Pagina(p)  \t Desplazamiento(d) \t Dirección Fisica\t\n");


// zero out the entire address space, to zero the unitialized data segment
// and the stack segment
   // bzero(machine->mainMemory, size);

// then, copy in the code and data segments into memory
    if (noffH.code.size > 0) {
         DEBUG('a', "Initializing code segment, at 0x%x, size %d\n",
              noffH.code.virtualAddr, noffH.code.size);
                                                                    //executable->ReadAt(&(machine->mainMemory[noffH.code.virtualAddr]),
              //noffH.code.size, noffH.code.inFileAddr);
    }
    if (noffH.initData.size > 0) {
```

```
          DEBUG('a', "Initializing data segment, at 0x%x, size
%d\n",
                noffH.initData.virtualAddr,
noffH.initData.size);

executable->ReadAt(&(machine->mainMemory[noffH.initData.virtua
lAddr]),
                noffH.initData.size,
noffH.initData.inFileAddr);
    }

}
```

5.- Archivo modificado: ./nachos/machine/addrSpace.h
Descripción:
Definicion del metodo swapIn para la clase Addrspace.
Código:

```
bool
AddrSpace::swapIn(int vpn)
{
    //file_name se agrego a machine
    printf("Haciendo swaping de %s\n",machine->swapFileName );
      OpenFile *swp = fileSystem->Open(machine->swapFileName);
//abrimos el archivo
    if(swp == NULL)
    {
        printf("\nswabFile no abrio\n");
        return false;
    }
    else
    {
            int direccionBaseDeMarco = stats->numPageFaults *
PageSize;
         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,vpn *
PageSize);

swp->ReadAt(&(machine->mainMemory[direccionBaseDeMarco]),PageS
ize,vpn * PageSize);
    }
    delete swp; //cerramos el archivo
```

```
    return true;
}
```

8.- Archivo modificado: ./nachos/machine/exception.cc
Descripción:
Modificación en el exception handler.
Código:

```
void
ExceptionHandler(ExceptionType which)
{
    int type = machine->ReadRegister(2);

    if ((which == SyscallException) && (type == SC_Halt)) {
     DEBUG('a', "Shutdown, initiated by user program.\n");
     interrupt->Halt();
    }
    /**************************************
    Practica 2
    **************************************/
    else if(which == PageFaultException)
    {
     //int vpn = machine->ReadRegister(BadVaddrReg)/PageSize;
     //currentThread->space->swapIn(vpn);
     if(stats->numPageFaults > NumPhysPages)
     {
          printf("\n\n::::::::::::::FAIL::::::::::::::\nNo  hay
suficientes marcos para terminar el proceso.\n\n");
          interrupt->Halt();
     }
     else
     {

     }
    }
    else
    {
     printf("Unexpected user mode exception %d %d\n", which,
type);
     ASSERT(FALSE);
    }
}
```

7.- Archivo modificado: ./nachos/machine/machine.h
Descripción:
Definición de la variable global para el nombre de archivo de intercambio..
Código:

```
/******************************
    Practica 2 Nombre del archivo de intercambio para el proceso.
    ******************************/
    char* swapFileName;
```

12. Describa los problemas que se presentaron en el desarrollo de la práctica y cómo se solucionaron.

Bueno nosotros en los problemas que tuvimos realizando la práctica fue principalmente al momento de intentar imprimir los distintos errores que se encontrarán con los diferentes marcos, fue más en las tomas de impresión, en lo cual nosotros incluyendo un factor principal de igualación que se incluyera con el fallo de página y al momento de incluir dicha falla se pudo tomar con un contador y la verificación de los diferentes datos, su memoria, su página, etc. Además de poder incluir una función en la cual nos generaría un apoyo entre la memoria principal y una externa para poder guardar dichos datos.