



Universidad de Costa Rica

Sede Rodrigo Facio

Facultad de Ingeniería

Escuela de Ciencias de la Computación e Informática

*Tarea 3*

*Documentación Externa*

Sistemas Operativos

CI – 1310

Estudiantes

Oscar Castro Espinoza - B11616 – Grupo 01

Abraham Vega Delgado - B27116 – Grupo 03

Profesor

Francisco Arroyo Mora

San Pedro, Montes de Oca, San José, 2014

## Introducción

En este documento, describimos qué se llevará a cabo en la Tarea 3. Para ser precisos, definiremos aquellos conceptos que implementaremos en NachOS posteriormente.

Para este proyecto nos centramos en la memoria del simulador del sistema operativo en cuestión. Nuestro objetivo principal será dotar de un sistema de memoria virtual a NachOS para que así ya no sea necesario tener todas las páginas de un programa en memoria principal al mismo tiempo.

Todo esto tiene como finalidad poder ejecutar programas de gran tamaño sin sobrecargar la memoria del simulador; además de comprender cómo funciona en realidad un sistema operativo. Las modificaciones principales se planean para los archivos `addrespace` y `exception`, como se nos indicó en el curso.

La idea de implementar estas funciones es facilitar las funcionalidades que ya posee el ambiente NachOS que se nos dio como base, dándole utilidades esenciales que tiene un sistema operativo real. Asimismo, se pretende dotarnos de práctica para entender de mejor manera algunos conceptos relacionados al campo de sistemas operativos.

## Desarrollo

En esta sección, resumimos el trabajo realizado durante el último laboratorio. Los primeros cambios se realizaron en el *addrespace*, comenzando con el constructor. A éste le agregamos un par estructuras condicionales para que lograra diferir entre memoria virtual y la memoria normal, *userprog*. Con este cambio aseguramos que el sistema operativo ejecute programas en ambas memorias.

Entre las funciones más relevantes para la aplicación, cabe mencionar las siguientes. Implementamos el método *secondChance* (*Figura 1*), encargado de buscar una página en memoria o en *TLB* que se pueda sacar. En otras palabras, saca una página de la memoria para introducir otra. Además, creamos una función que se encarga de mapear los casos para manejar los *pagefault exception*. Esta función la llamamos *MemoriaVirtual*, y ella se encarga de determinar cómo actuar en cada uno de los escenarios posibles (*Figura 2*). Asimismo, programamos dos funciones relacionadas al archivo *swap*, *leer* (*figura 3*) y *escribir* (*figura 4*). Gracias a ambas fue posible simular el funcionamiento de un swap en un sistema operativo.

```

// Método que ayuda a la tabla de paginación cuando ésta está llena.
int AddrSpace::secondChance(int vpn)
{
    bool continuar = true;
    int indice = ult;
    // Buscamos las páginas que tienen el bit de use válido.
    while(continuar)
    {
        // Verificamos el estado del bit de use.
        if(pageTable[vectorP[indice]].use)
        {
            // Lo definimos en false.
            pageTable[vectorP[indice]].use = false;
            ++indice;
            // Lo hacemos como un vector circular.
            indice = indice % NumPhysPages;
        }
        else
        {
            ++ult;
            ult = ult % NumPhysPages;
            ult = false;
        }
    }
}

```

Figura 1  
Parte del método secondChance

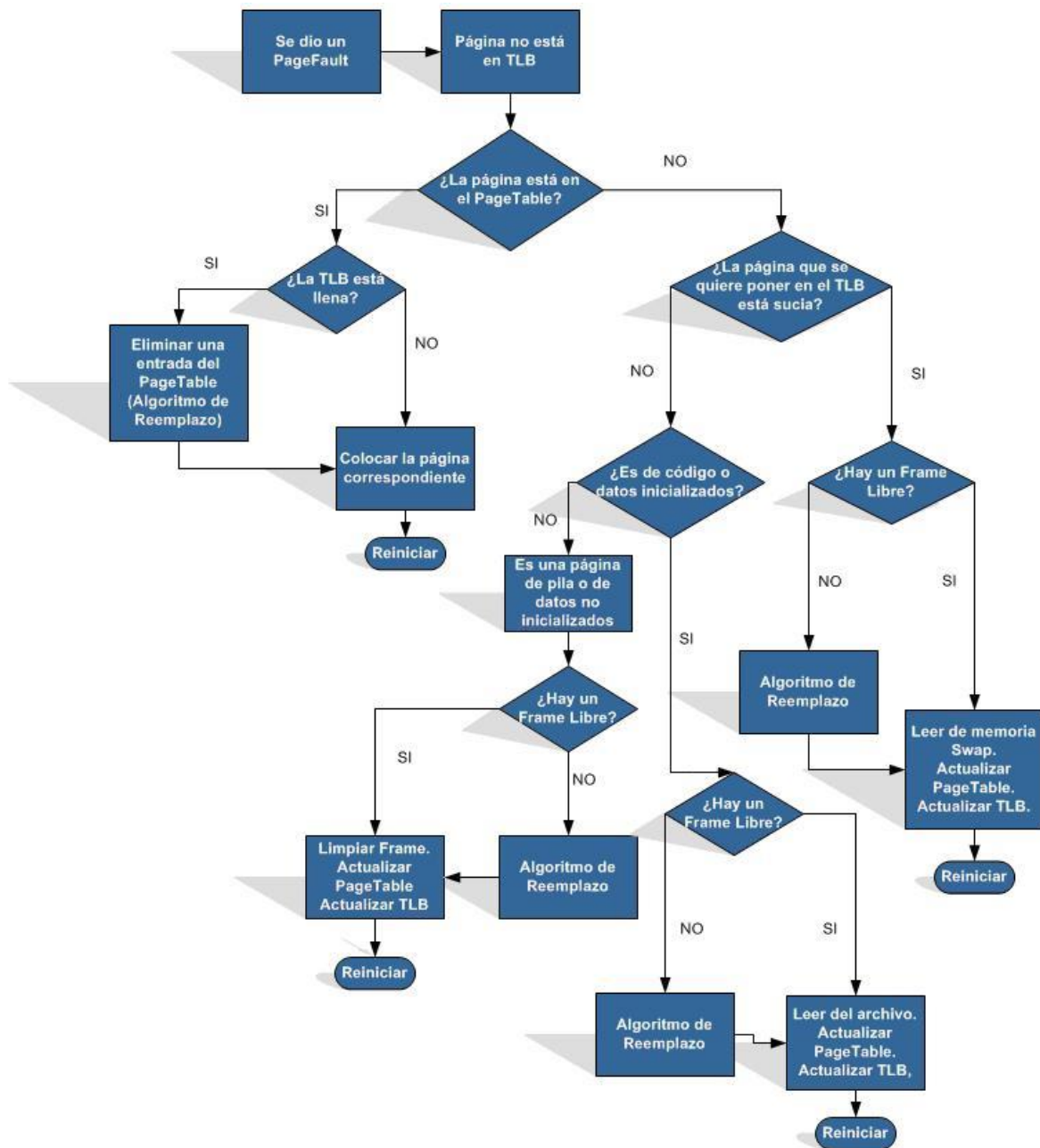


Figura 2  
Control de Escenarios

```

// Método encargado de leer del SWAP y de borrar de memoria esa página.
void AddrSpace::leer(int page, int vpn)
{
    // Abrimos el archivo SWAP.
    OpenFile* swap = fileSystem->Open("Swap");

    int i = 0;
    bool continuar = true;
    // Buscamos la página deseada.
    while(continuar)
    {
        if(vectorS[i] == vpn)
        {
            // Termina si encuentra la página en el SWAP.
            continuar = false;
        }
        else
        {
            i++;
        }
    }

    swap->ReadAt(&(machine->mainMemory[page*PageSize]), PageSize, i*PageSize);
    // Limpiamos el bitmap del SWAP.
    MiMapaSwap->Clear(i);
    // Invalidamos esa posición del vector auxiliar.
    MiMapaSwap[i] = -1;
    delete swap;
}

```

*Figura 3*  
*Método de Leer para el Swap*

```

// Método encargado de guardar en el SWAP la página deseada.
void AddrSpace::escribir(int page , int pos)
{
    // Abrimos el archivo SWAP.
    OpenFile* swap = fileSystem -> Open("Swap");
    // Guardamos en SWAP la página deseada.
    swap->WriteAt(&(machine -> mainMemory[page*PageSize]), PageSize, pos*PageSize);
    delete swap;
}

```

*Figura 4*  
*Método de Escribir para el Swap*

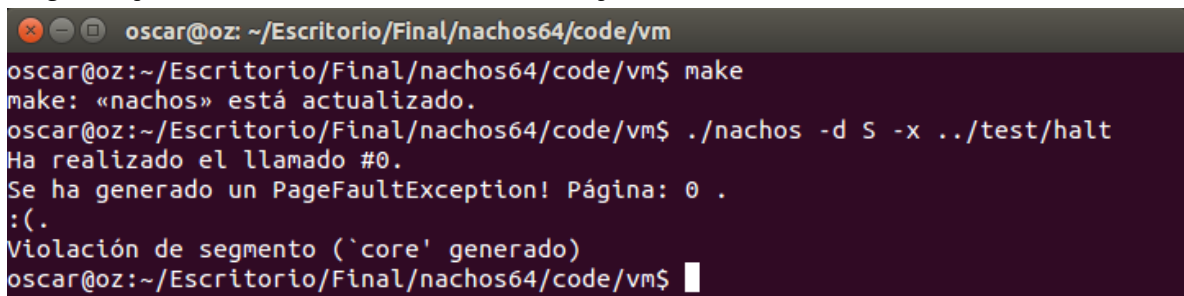
## Manual de Usuario

### Compilación

Para lograr compilar nuestro proyecto, bastará con posicionarse en el directorio nachos64/code/vm. Estando ahí, es necesario ingresar el siguiente comando en la terminal de Linux, *make*. Éste hará que todos los archivos que necesiten ser compilados lo sean.

### Ejecución de Pruebas

Para lograr ejecutar algunas pruebas sobre nuestro proyecto, bastará con posicionarse en el directorio nachos64/code/vm. Estando ahí, sólo se necesitará ingresar el siguiente comando en la terminal `./nachos -d S -X ../test/NombreDeLaPrueba`. A continuación se adjunta un par de imágenes que muestran lo sucedió al correr un par de éstas.



```
oscar@oz: ~/Escritorio/Final/nachos64/code/vm
oscar@oz:~/Escritorio/Final/nachos64/code/vm$ make
make: «nachos» está actualizado.
oscar@oz:~/Escritorio/Final/nachos64/code/vm$ ./nachos -d S -x ../test/halt
Ha realizado el llamado #0.
Se ha generado un PageFaultException! Página: 0 .
:(.
Violación de segmento ('core' generado)
oscar@oz:~/Escritorio/Final/nachos64/code/vm$
```

Figura 5

*Método de Escribir para el Swap*

### Problemas

Desde el proyecto anterior, no nos fue posible solucionar algunos problemas que generan violaciones de segmento. Por esta razón, no nos fue posible terminar las pruebas con éxito. A pesar que para esta prueba se detectó el *pagefault* donde debía hacerse, el programa colapsa.

ACLARACIÓN, la figura 2 no pertenece a nuestra autoría, fue recuperada de la siguiente página <http://os.ecci.ucr.ac.cr/ci1310/pagefault.jpg>.