Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

CI-1310 Sistemas Operativos
Grupo 02|1
II Semestre

Tarea programada NachOS#1

Profesor:
Francisco Arroyo

Estudiantes:
Dennis Abarca Quesada | A70014 | grupo 01
Steven Rojas Lizano 1 | A75623 | grupo 02

October 30, 2016

Contents

1 introducción **1**

2 Objetivos **1**

3 Descripción **2**

4 Pruebas **3**

5 Manual de usuario **4**
5.1 Requerimientos 4
5.2 Restricciones 4

6 Bibliografía **5**

1 Introducción

En esta tarea programada se realizó la implementación de los llamados al sistema de nachOS, en los cuales se realizaron modificaciones a los archivos de addresSpace, system, exception y machine.

2 Objetivos

- Leer y entender la parte del sistema operativo NachOs que se encuentra en la carpeta descargable.
- Implantar el manejo de excepciones y llamados al sistema.
- Implantar multiprogramación.
- Implantar programas de usuario multi-hilos.

3 Descripción

1. Implantar el manejo de excepciones y llamados al sistema. Se deben soportar todos los llamados al sistema definidos en "syscall.h". Presentamos una rutina en ensamblador "syscall" que provee la manera de invocar un llamado al sistema desde una rutina C (Unix tiene un método similar, intente "man syscall"). Usted necesita completar la parte 2 de esta asignación con el fin de probar los llamados al sistema "exec" y "wait"

2. Implantar multiprogramación. El código que presentamos le restringe a solo poder correr un programa de usuario a la vez. Usted necesita:

   (a) Encontrar la manera de asignar los marcos de memoria física de tal manera que varios programas de usuario puedan ser colocados en la memoria principal a la vez. (ver "bitmap.h")

   (b) Proveer una manera de copiar datos desde/hacia el kernel desde/hacia el espacio de direcciones virtual del usuario (ahora las direcciones que el programa del usuario 've' no son las mismas que el kernel 've')

3. Agregar sincronización a las rutinas que crean e inician el espacio de direcciones, de tal manera que puedan ser accedidas concurrentemente por múltiples programas. Note que "scheduler.cc" ahora guarda y recupera el estado de la máquina en los cambios de contexto. Es deseable tener algunas rutinas como "cp" y "cat" de Unix, y utilizar el shell que proveemos en el directorio "test" para verificar el manejo de llamados al sistema y la multiprogramación

4. Implantar programas de usuario multi-hilos. Implante los llamados al sistema "fork" y "yield", que le permita al usuario llamar a una rutina en el mismo espacio de direccionamiento, y hacer ping pong entre los hilos (Ayuda: necesita cambiar la manera actual del kernel para asignar memoria en el espacio de direcciones del usuario para cada una de las pilas de los threads)

5. (Extra, por 5%)La versión actual del llamado al sistema "exec" no ofrece ninguna manera para que el usuario pueda pasar parámetros o argumentos al nuevo espacio de direcciones creado. Unix permite ésto, por ejemplo, se pueden pasar argumentos en la línea de comandos al nuevo espacio de direcciones. Implantar esta funcionalidad del "exec"

4 Pruebas

5 Manual de usuario

5.1 Requerimientos

El sistema posee los siguientes requerimientos para su correcto funcionamiento

- Sistema operativo Fedora
- Arquitectura 64 bits
- Ambiente NachOS
- Compilador: GNU GCC compiler

5.2 Restricciones del programa

- El sistema solamente puede ser ejecutado desde la terminal.
- El sistema solamente trabaja con archivos codificados en UTF-8 y codigos fuente del lenguaje C.

6 Bibliografía

[1] Silberchatz, Abraham, Galvin, Peter & Gagne, Greg. *Operating Systems Concepts.* Novena edición, Addison Wesley Publishing Co., Mass., 2013