**REPORTE EXAMEN 1ER PARCIAL**

***Problema.*** Describe el proceso para agregar una nueva syscall en xv6, e implementa una diferente a las agregadas en clase.

Para poder agregar una syscall, se deben modificar al menos 4 archivos del código de xv6:

1.-syscall.h

Aquí se define una macro con el nombre de la syscall y se le asigna un número que es su identificador en una tabla de syscalls.

2.-syscall.c

Se coloca el prototipo de la syscall usando extern para poder definirla en otro archivo, aquí se tiene un arreglo de apuntadores a función, el cual contiene todas las syscall y debe usarse la macro definida en syscall.h

2.1-Para implementar la funcion de la syscall existen dos archivos, sysproc.c y sysfile.c, el último se usa para las syscalls que usen el filesystem y en el otro archivo están las syscalls restantes. Las syscalls utilizan servicios de dos diferentes archivos, proc.c para servicios que no utilicen el filesystem y fs.c para los que sí lo utilicen. Ambos prototipos están definidos en defs.h

3.-usys.S

Ahora se debe definir una macro con el nombre de la syscall, esta macro crea un código en ensamblador que utiliza el numero definido en syscall.h para guardarlo en el registro eax, el cual se usa como índice en el arreglo de syscalls definido en syscall.c, después se crea una interrupción con la instrucción int

4.-user.h

Aquí debemos poner el prototipo de nuestra syscall usando solo el nombre, y este archivo sirve para poder usar la syscall en el espacio de usuario.

**AGREGANDO UNA NUEVA SYSCALL**

Para este ejercicio se agregó una syscall llamada numcalls, esta syscall devuelve el número de veces que se ha llamado a las syscalls. Usamos una variable compartida para contar el número de veces que se ha llamado a las syscalls, la variable se llama syscallnum, debido a que es una variable compartida y puede ser usada por varios procesos a la vez, debemos de protegerla para que no existan inconsistencias en la lectura o escritura de esta. Para esto usamos el mecanismo de locks que trae xv6, las modificaciones son las siguientes.

En syscall.h se agregó:

#define SYS\_numcalls 22 */\*El número de nuestra syscall\*/*

En syscall.c:

#include "spinlock.h" */\*Archivo para usar locks\*/*

struct spinlock numslock; */\*Estructura necesaria para proteger la variable\*/*

int syscallnum=0; */\*Para guardar el número de llamados a syscall\*/*

void numinit() */\*Función que inicializa el lock\*/*

{

initlock(&numslock,"num");

}

int

sys\_numcalls(void) */\*Implementacion de nuestra syscall, en este ejemplo*

*no se usó extern\*/*

{

int num; */\*Variable que guarda el valor de syscallnum\*/*

acquire(&numslock); */\*Bloquea el acceso a syscallnum\*/*

num=syscallnum; */\*Lee la variable\*/*

release(&numslock); */\*Libera el acceso a syscallnum\*/*

return num; */\*Devuelve el valor de syscallnum\*/*

}

/\*En la estructura static int (\*syscalls[])(void)\*/

[SYS\_numcalls] sys\_numcalls, */\*Se guarda la syscall en el arreglo de*

*syscalls\*/*

En usys.S:

SYSCALL(numcalls) */\*Macro que genera la interrupción a nuestra*

*syscall\*/*

En user.h:

int numcalls(void); */\*Funcion para usarla en el espacio de usaurio\*/*

En main.c:

*/\*Dentro de main()\*/*

numinit(); *//inicializa syscallnum lock*

Ahora para poder probar nuestra nueva syscall, creamos un programa que imprime el número de llamadas a syscalls, el programa se llamasyscallnum y su código es el siguiente:

#include "user.h"

#include "types.h"

int main(){

printf(1,"El numero total de llamadas a syscall es: %d\n",numcalls());

exit();

}

En el Makefile agregamos \_syscallnum/ en U\_PROGS y ya podemos usar nuestro programa.

Resultado:

