primer labo donde metemos mano sobre codigo fuente

xv6: so educativo que lo corremos con un emulador

hacer systems calls para interactuar con el vga, para tocar el espacio de memoria y confirgurar el vga en el modo grafico

corre en modo texto: pasar de modo texto a modo grafico

-los espacios de direccion se solapan.

vga esta configurada para arrancar en modo texto, voy a tener que escribir unos registros para pasar a modo grafico o a modo texto

trampa: como estan solapados al pasar de un modo a otro tener cuidado de no perder toda la informacion

se reutiliza una direccion de memoria para reescribir varios registros

-tareas: descargar el repo de xv6(codigo)

tener qemu (para poder ejecutarlo)

en los videos: ejemplo de como crear una funcion en el espacio de salida

1- implementar un pie de pantalla escribiendo en el lugar correspondiente de memoria desde el espacio de kernel

2-extender con funciones que permiten cambiar de modo texto a modo grafico(las llamaremos system calls) y usarlas desde el momento(adentro del codigo las puedo buscar con grep)

3-implementar la syscall en si misma: la que cambia de modo cuando recibe 0 o 1 cambia de un modo a otro

syscall q

4- programa de usuario que use las syscalls que dibuje algo en pantalla

extra: modularizar

syscall

ayudas:

mirar el makefile de xv6 porque hace varias cosas a la vez, por una lado compila y por otro genera una imagen de disco con todos los programas de usuario que uno va a querer usar, agregarlos a la parte del makefile que es la genera la parte de imagen del disco

grep para encontrar como aparece y donde

-el resgisttro de direcciones y de datos hay que escribir donde vamos a escribir

para el mapeo de memoria usar macro?

**introduccion a xv6, parte1**

es un so, es una reimplementacion de unix

la idea fundamental es separar el espacio de usuario y el espacio de kernel(encargado de interactuar directamente con el hardware, file system. Para brindar una capa de abstraccion para que el SO sea robusto ante implementaciones de programas o procesos por parte del usuario. Lo que hace el espacio de kernel es brindar una api o interfaz que son un conjunto de funciones que se llaman syscall que el usuario puede utilizar para poder realizar algunas acciones con el procesador, etc.

system calls:

-shell: proceso de usuario

aparte del proceso inicial que es el init, arranca todo el so. corre automaticamente el proceso shell, se queda esperando una serie de instrucciones por parte del usuario para ser ejecutado

-cat:

para mostrar el contenido de un archivo

estos procesos viven en el espacio de usuario y se ejecutan en el espacio de usuario

scall de xv6:

-fork(): create process

-exit(): terminate current process

-wait(): wait for a child process

-kill(pid): terminate process pid

-getpid(): return current process’s id

-sleep(n): sleep for n seconds

-exec(filename, \*argv): load a file and execute it

-sbrk(n): Grow process’s memory by n bytes

-open(filename, flags): open a file, flags indicate read/write

-read(fd, buf, n): read n bytes from an open file into buf

-write(fd, buf, n): write n bytes from an open file into buf

-close(fd): release open file fd

-dup(fd): duplicate fd

-pipe(p): create a pipe and return fd’s in p

-chdir(dirname): change the current directory

-mkdir(dirname): create a new directory

-mknod(name, mayor, minor): create a device file

-fstat(fd): return into about an open file

-link(f1, f2): create another name (f2) for the file f1

-unlink(filename): ??

min6:09

p descargar

$git clone <https://github.com/mit-pdos/xv6-public>

p compilar tenemos que tener instalado el qemu

$sudo apt-get install qemu

ahora podemos entrar a la carpeta con el codigo fuente y compilar:

$cd xv6-public ///adentro de la carpeta

$make qemu //ejecuta un makefile y lanza la interfaz grafica una vez que compila

.la nueva ventana es la interfaz grafica del so

tengo dos ventanas(qemu novec para que no se lance la parte grafica)

lo que escribo en la terminal se escribe y ve tambien en la interfaz grafica

-qemu: es la maquina virtual sobre la cual corre esta maquina virtual

**shortcut:**

-listar todos los procesos que estan corriendo en xv6

<CTRL-p>

-salir de xv6

<CTRL -a> x

**introduccion a xv6, parte2**

https://course.ccs.neu.edu/cs3650/unix-xv6/index.html

user.h---> declara varias system calls: estamos definiendo una interfaz para el usuario

echo.c---> para crear la funcion de usuario echo

-ejercicio: crear un proceso en el espacio de usuario que utilice una syscall (uptime) que muestra cuantas interrupciones de tick del reloj han ocurrido desde el inicio de un so