Trabajo Práctico Especial
Segundo Cuatrimestre 2025

Objetivo

Implementar un kernel que administre los recursos de hardware de una computadora y muestre características del modo protegido de Intel.

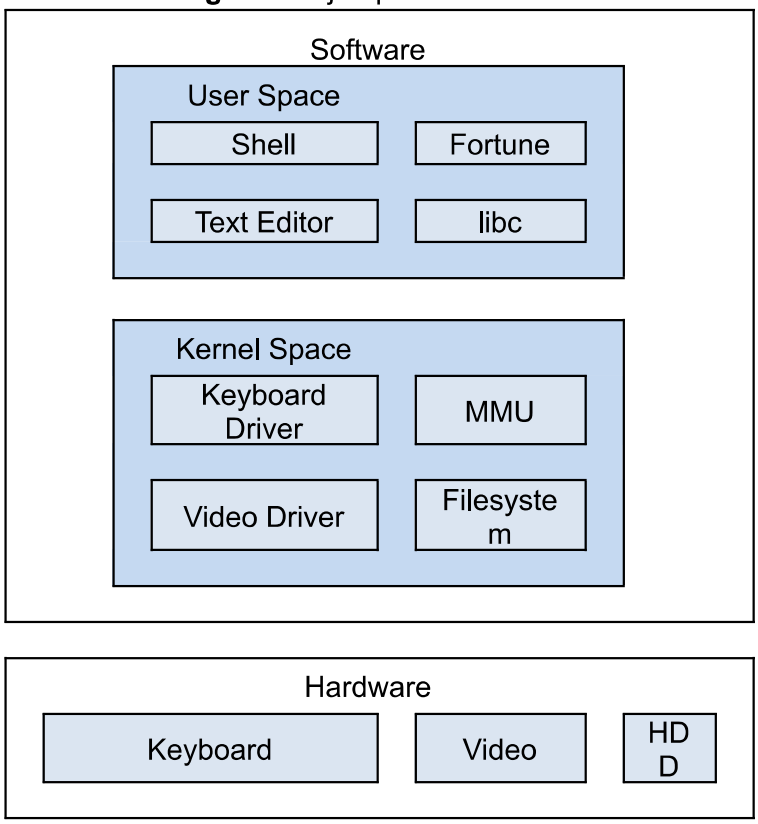
Enunciado

Implementar un **kernel** booteable por Pure64, modificado y provisto por la cátedra. El mismo debe administrar los recursos de hardware y proveer una API para que aplicaciones de usuarios puedan utilizar estos recursos.

Se deben definir dos espacios claramente separados, uno **kernel space** y el otro **user space.** El espacio del kernel interactuará directamente con el hardware mediante drivers, mientras al mismo tiempo proveerá funciones al user space. El espacio de usuario no debe acceder por ninguna razón directamente al hardware, sólo a través del espacio de kernel. En la **Figura 1** se puede ver un ejemplo de los módulos que conforman el espacio de usuario y los módulos que conforman el espacio de kernel.

Se deberá definir una **API** con la cual el espacio de usuario puede acceder a las funciones del kernel. Este acceso deberá ser implementado a través de la interrupción de software 80h, ya que estos dos módulos se encuentran en distintos espacios de memoria. Dicha API deberá basarse en la API de Linux, que está comprendida por funciones cómo **read** y **write.**

Figura 1: Ejemplo de módulos.



Además se deberá definir un set de funciones para interactuar con dicha API. Deberá ser el equivalente en Linux a la biblioteca estándar de C, en la cual se deberá basar. Por ejemplo, deberá contar con funciones como **scanf**, **printf**, **putChar**, **getChar**.

La implementación del sistema se detalla a continuación.

Intérprete de comandos

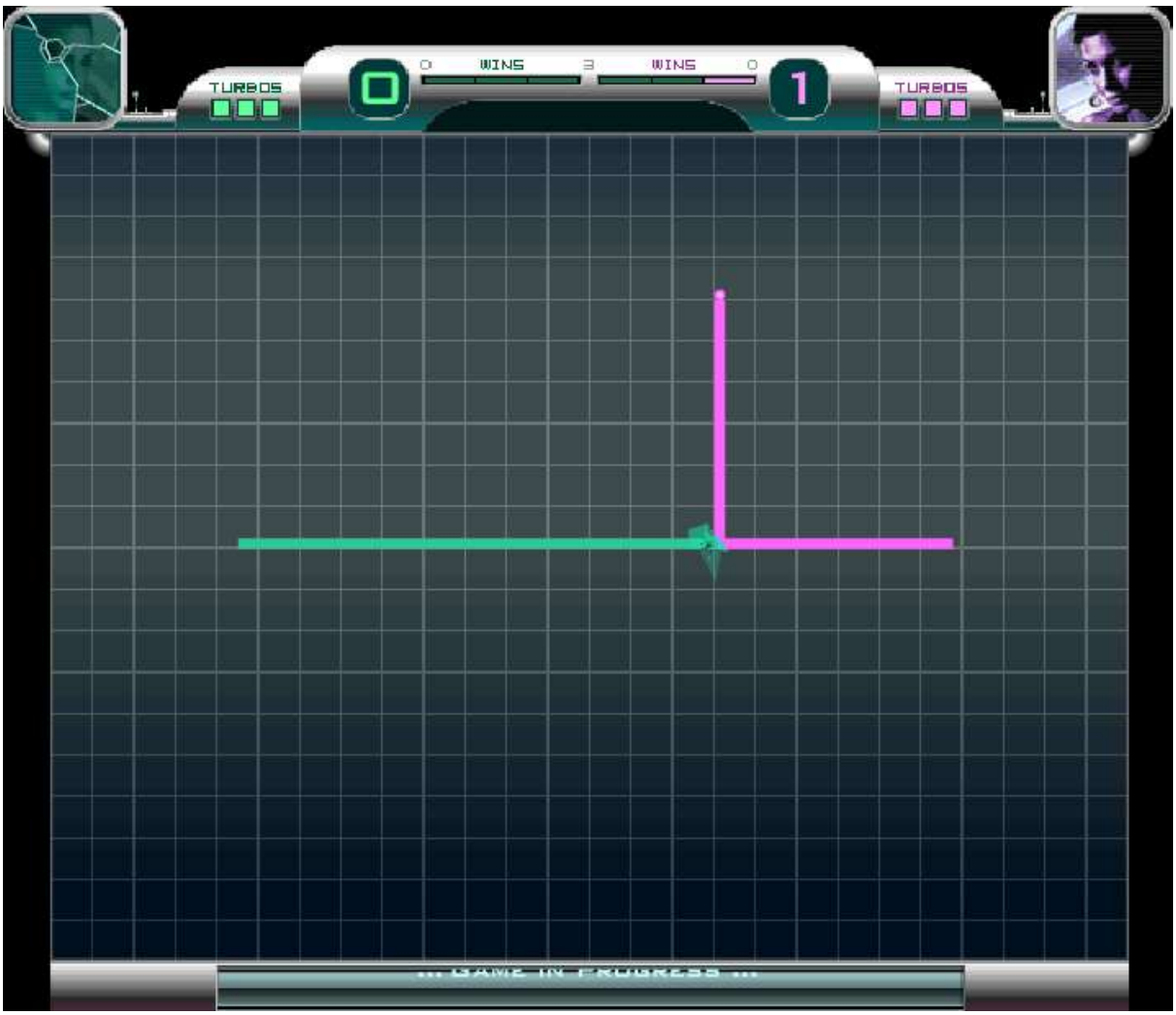
El cual, a través de los comandos de la shell, demuestre el funcionamiento del kernel. Deberá tomar dichos comandos por entrada estándar y mostrar el resultado de la ejecución por salida estándar. El intérprete deberá interactuar con el sistema mediante un modelo de Terminal. Este intérprete de comandos deberá siempre desplegar su prompt para ingreso de nuevos comandos.

Funcionalidades

- Una función de ayuda, que muestre los distintos programas disponibles.
- Intérprete de comandos, tipo **Shell** que desencadene las distintas funcionalidades.
- Módulo/comandos para desplegar la hora del sistema
- Luego de que el sistema arranca, queda el prompt en espera a que se ingrese qué módulo se desea correr.
- Se deberá disponer de una funcionalidad que permita obtener el valor de los todos los **registros del procesador** en cualquier momento que se desee.
- Módulo/comando que permita graficar en modo video el juego “tron”.

https://www.crazygames.com/game/fl-tron

- Implemente la selección de 1 jugador en pantalla o que sean 2 los jugadores en pantalla (simultáneamente sobre un único escenario, ambos jugadores compiten por quién perdura más tiempo / más recorrido).
- Use un conjunto de teclas para el movimiento de los jugadores y manteniendo en pantalla un puntaje.
- Deberá emitir un sonido en momentos que considere oportunos del juego.
- **Simplifique los gráficos**, no es requerido detalles de texturas, mas puede implementar el juego según su propio criterio visual.
- **Simplifique el juego todo lo que usted considere apropiado.** Ejemplo:los niveles pueden consistir en unicamente 1 velocidad o puede ir escalando según avanza. El terreno puede ser siempre plano o puede insertar obstaculos. Idea: si quiere trabajar con física del terreno, puede colorearlo, y que tal variable implique diferentes “densidades” que cambian la velocidad del jugador, lo cual puede ser un agregado de generación de diferentes dificultades al jugador. Use su criterio.



- Haga todas las mejoras en la jugabilidad que considere apropiadas, ya sea en lo que respecta a velocidad del juego, visuales, u otros aspectos del ejemplo.

- Comando que permita agrandar o reducir el tamaño del texto de la pantalla (se refiere a dentro de la shell).
- Cualquier otra función que el grupo considere apropiada para mostrar el funcionamiento interno del sistema.

  Los requerimientos implican la implementación de drivers de teclado y video.

Funcionalidades de Excepciones

- El kernel deberá poder manejar dos tipos de **excepciones**: división por cero y código de operación inválido. Ante estas excepciones se deberá desplegar información sobre el tipo de error, instruction pointer y todos los registros del procesador en el momento del error.
- El sistema debe poder recuperarse posteriormente de la excepción, regresando a la Shell.
- Rutinas para verificar el funcionamiento de las excepciones antes descritas.

Funcionalidades de Benchmarking

Además de las anteriormente mencionadas características, se deben agregar las siguientes funcionalidades que permitan tomar parámetros de benchmark, por ejemplo:

- FPS
- Rendimiento/tiempos de procesamiento (por ejemplo en cálculos con uso de Floating Point)
- Tiempos de acceso a hardware.

Estas funcionalidades deberán ser probadas, comparadas y analizadas en el informe para al menos 4 casos diferentes:

- QEMU
- Hardware físico en al menos 2 máquinas diferentes (dar detalles del HW utilizado).
- Virtual Box

  Pueden tomar ideas de lo que hace https://www.memtest86.com/

Desarrollo, implementación y presentación

El sistema deberá ser implementado para una Arquitectura Intel de 64 bits en **Long Mode.** La entrada a ese modo será provista por un módulo implementado por la cátedra y debe correr en una PC física real.

El desarrollo de sistema se realizará integramente en QEMU. En la instancia de presentación, el grupo debe venir preparado para realizar la demostración de su SO corriendo en **QEMU.**

Por otro lado, en la instancia de presentación, se creará una imagen en un dispositivo removible y se **booteará en una PC física real**. En caso de bootear exitosamente el SO en la máquina física, será en esa máquina donde se realizará la exposición y demostración del

sistema implementado. En caso de no bootear o de bootear pero tener problemas en el driver de video, se realizará la presentación en QEMU.

Compilación

El script de compilación del SO implementado debe contemplar un **argumento** el cual permita realizar el armado de la imagen para QEMU o alternativamente, para un determinado HW físico que oportunamente la cátedra proporcionará. Es por ello, que el driver de video, debe contemplar el correcto funcionamiento para dibujar en pantallas de distintas resoluciones.

Fuentes

Cómo base del trabajo práctico, descargar los archivos del siguiente repositorio:

```
https://github.com/alejoaquili/x64BareBones
```

Integrantes

El grupo de trabajo podrá contar con un máximo de **tres** (3) integrantes.

Calificación

La nota del trabajo práctico consta de la evaluación de los entregables más la **demostración** de su funcionamiento en un coloquio oral. En dicho coloquio se podrá consultar a cualquier integrante cualquier porción de código cuál es su implementación y funcionamiento. Además se podrán consultar conceptos teóricos de la materia.

A la hora de la presentación, todos los miembros deberán estar presentes, salvo causa de fuerza mayor, en la cual, el integrante faltante deberá acreditar la documentación que justifique la falta. Dicho integrante luego deberá rendir el coloquio oral en la fecha que se acuerde con la cátedra.

En el caso de que se deba **re-entregar** el TPE por no estar aprobado en primera instancia, se penalizará a los integrantes del grupo con una **disminución de 1 (un) punto** sobre la nota final. Es decir, será necesario un 6 (seis) para aprobar el trabajo práctico en instancia de recuperación.

En el caso de que al momento de la entrega, el TPE no tenga inconvenientes de booteo ni de video en la PC física designada por la cátedra para tal fin, sumará 1 (un) punto sobre la nota final.

La califiación resultará de los siguientes conceptos:

Nota TPE	=	Instancia Presentación	+	Revisión del código entregado	+	Revisión del Informe entregado	+	No se pide reentrega / ni correcciones	+	PC física
		(5.5 ptos)		(1.25 ptos)		(1.25 ptos)		(1 pto)		(1 pto)

Entregables

Cada grupo deberá entregar por email, un archivo ZIP (compartido desde el google drive, puesto que si intentan adjuntar directo el archivo, es bloqueado por cuestiones de seguridad) con lo siguiente:

- **Código fuente** (sin los binarios).
- **Manual de usuario.** Que detalle los comandos implementados y toda otra información que considere pertinente para el uso de su shell.
- **Informe.** Que explique el diseño elegido con sus justificaciones, pros, contras. Aspectos técnicos y toda otra información que resulte relevante al desarrollo del trabajo.
- Correspondiente **hash md5**, en el cuerpo del mail.

El archivo debe estar nombrado cómo gNN-X-Y-Z.zip, donde X, Y, Z son los legajos de los participantes y NN es el número del grupo (NN = 01, 02, 03,…). Tener en cuenta que el hash md5 es la firma del grupo sobre el entregable. En caso de tener problemas con el envió del TPE, enviar el hash directamente.

El link del archivo compartido con todo el material deberá ser enviado en forma digital el día de la entrega vía email a la casilla:

arq-entrega-tpe@googlegroups.com

Consideraciones

- El tipo de diseño y la forma de implementación serán discutidos entre el grupo y la cátedra durante las clases de laboratorio, dejando la posibilidad de modificar éste enunciado escrito, previo acuerdo entre el docente y los integrantes del grupo.
- Para la evaluación se tendrá en cuenta no sólo el **funcionamiento** del programa sino también la **calidad del software y la documentación** escrita pedida.
- Cualquier funcionalidad agregada al TPE por sobre lo pedido deberá estar documentada apropiadamente.
- Cualquier código encontrado en Internet u otra fuente debe ser consultado con la cátedra previo a la inclusión en el TPE. Toda inclusión de código de terceros deberá ser documentada con los enlaces o referencias correspondientes a su origen. Deben estar sumarizados en el informe de la cátedra.
- Cualquier aclaración oral a cargo de la cátedra con respecto al enunciado del TPE tiene la misma validez que el enunciado escrito.
- Los distintos grupos podrán consultar y discutir diseños entre sí, pero la implementación deberá ser propia de cada uno. Cualquier detección de plagio es susceptible de sanciones académicas de acuerdo al reglamento de la Universidad.
- Todos los trabajos corren sobre la misma arquitectura y cuentan con el mismo template, por lo tanto existirán muchos códigos equivalentes o similares. La cátedra tendrá esta situación en cuenta.

Fechas de entrega y defensa

La **entrega del TPE completo con el manual de usuario e informe** es hasta el:

Miercoles 5 de noviembre a las 00:00hs

Via correo electrónico dirigido al correo de la cátedra.

Habrá una **defensa presencial** para todos los grupos. Hay 4 fechas, en los días:

Miércoles 5 de noviembre a las 16hs

Jueves 6 de noviembre a las 16hs

Miércoles 12 de noviembre a las 16hs

Jueves 13 de noviembre a las 16hs

Oportunamente se publicará un horario y día de presentación para cada grupo.

Terminada esta instancia, se informará aquellos grupos que tengan que hacer una nueva entrega (**recuperatorio TPE**).

La fecha de entrega del **TPE recuperatorio** será hasta el:

Miercoles 19 de noviembre a las 00:00hs

Vía correo electrónico.

La defensa (de la reentrega) será los días:

Miercoles 19 de junio a las 16:00hs

Jueves 20 de junio a las 16:00hs