Universidad Nacional de La Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
Grupo 17 – Driver PCI

Sistemas Operativos
Código 625

Trabajo Práctico
Driver PCI

Grupo 17

Alumnos

Apellido	Nombre	D.N.I.	E-mail
Magliola	Adrian Matias	32.438.367	adrian.magliola@gmail.com
Suc	Dario	30.067.719	dariosuc@hotmail.com
Baini	Sebastian	28.596.449	sebasbaini@gmail.com
Romano	Mercedes	32.961.251	romanomercedes@gmail.com

FECHA DE PRESENTACIÓN: 01/11/2012
VERSIÓN: 1.0.0

Contenido

1. Desarrollo 3
1.1 Objetivos 3
1.2 Estructura 3
1.3 Practica 4
1.3.1 Introducción a la Práctica 4
1.3.2 Resumen de nuevos agregados 4
1.3.3 Nuevas Funciones 5
1.3.4 Casos De Prueba. 15
1.4 Archivos Fuentes 17
1.4.1 Comentarios por funciones y archivos 17
1.4.2 Comentarios en las funciones 17
1.4.3 Nomenclatura Sodium 17
2. Conclusiones 17
2.1.1 Problema con archivos USB 17
2.1.2 Problema con el SVN 17
2.1.3 Problema con la memoria de SODIUM 17
2.2. Conclusión 17
3. Evolución de TP (Mejoras) 18
4. Apéndice 18
4.1 Desarrollo de drivers específicos 18
4.1.1 Consideraciones especiales para el Driver USB 19
4.2 Desarrollo de drivers específicos 19

1. Desarrollo

1.1 Objetivos

El objetivo es dotar a SODIUM de la capacidad de gestionar cualquier dispositivo que se conecte a una computadora a través del bus PCI.
Para alcanzar este objetivo se desarrollará un driver que le permita a SODIUM detectar cualquier dispositivo que se conecte a la computadora mediante el bus PCI, realizar su configuración básica y administrar tanto la comunicación de los mismos con la CPU como el uso de los recursos compartidos de la computadora, por ejemplo la memoria RAM.

El driver PCI deberá cumplir con las siguientes funciones:

1. Realizar un polling de los dispositivos conectados al bus PCI.
2. Acceder a los registros de configuración de cada dispositivo y guardar esta información en estructuras.
3. Armar una vector con todas las estructuras de los dispositivos en memoria RAM.
4. Habilitar los dispositivos mediante sus registros bases (BAR).
5. Iniciar el driver específico para un dispositivo en particular.
6. Dotar a los drivers específicos de funciones generales provistas por el driver PCI.

Alcance: Nos proponemos alcanzar hasta la función N°6, con la salvedad que en la función 5 intentaremos centrarnos integrar el driver específico para le manejo de puertos USB.

1.2 Estructura

En el comienzo del desarrollo del driver para el bus PCI, se trato la detección de dicho bus en la ultima etapa del la inicialización del sistema operativo SODIUM. En donde también se realiza la inicialización de la controladora PCI.
El controlador PCI recorrerá los 256 buses posibles, por cada bus los 32 dispositivos que pueden estar conectados a éste y por cada dispositivo sus potenciales 8 funciones. Este controlador realizará ciertas operaciones con los registros del bus PCI para obtener una dirección específica de 32 bits, la cual será utilizada para acceder al bus de un dispositivo en particular. Esta dirección se denomina CONFIG_ADDRESS y permite la identificación del dispositivo que se encuentra conectado al bus y la especificación del mismo.
Con esta dirección del dispositivo, se leerá una estructura de 256 bytes. Esta estructura contiene los datos necesarios para iniciar una interacción con el dispositivo, logrando así, escribir en sus registros los valores para la inicialización.
Estos 256 bytes, están compuestos por una región cabecera de 64 bytes común a cualquier clase de dispositivo PCI y otra región dependiente del dispositivo de 192 bytes.
Se desarrollaron las funciones necesarias para la identificación de los distintos dispositivos conectados al bus PCI, inicialización, comunicación (lectura y escritura), manejo del áreas de espacio de memoria o espacios de puerto entrada/salida que el dispositivo solicite utilizar.
Son los registros base, BARs (Base Address Register), son los encargados de informar a la BIOS del número y tamaño de los espacios de memoria o de entrada/salida necesarios y de identificar una zona de memoria del sistema mapeada sobre un determinado dispositivo PCI.
Por ultimo, se agrego un comando capaz de mostrar la información relativa a los distintos dispositivos PCI conectados y sus detalles.

1.3 Practica

1.3.1 Introducción a la Práctica

Para introducirnos en la codificación intentamos identificar cuales eran las distintas evoluciones que teníamos que seguir para poder construir el driver:
Dentro de las evoluciones podemos destacar:

N° Iteracion	Descripcion	Acciones
1°	Desacoplar los archivos de USB. y dejar los archivos PCI.C y PCI.H	- Se borraron todas las referencias USB en SODIUM: funciones, variables y syscalls. - Se borraron los archivos USB: usb, usbhub, uhci, usb_teclado y usbtrans. - Se blanquearon los archivos pci.h y pci.c
2°	Detectar el Bus PCI	Se incluyo la función encargada de detectar la presencia de un bus PCI.
3°	Recorrer Bus PCI	Una vez detectado el bus, se incluyeron funciones para recorrer todos los posibles buses, dispositivos y funciones de un sistema PCI, los cuales se deberían guardar en una estructura.
4°	Configurar un dispositivo	Se agregaron funciones para una vez detectado un dispositivo configurarlo, eso incluye: - Habilitar el dispositivo. - Llamar al driver específico según el tipo de dispositivo. - Agregar la información de los dispositivos en memoria (vector de estructuras)
5°	Funciones generales	Se desarrollaran funciones generales para proveer servicios a los driver específicos de los dispositivos. O sea, que los programadores de los distintos dispositivos PCI utilicen funciones generales que provee el driver PCI.

1.3.2 Resumen de nuevos agregados

-Paquetes al SDK **No aplica**
-Bibliotecas nuevas **No aplica**
-Cambios en SODIUM

Se editaron los siguientes archivos: /usr/bin/lspci.c, include/kernel/drivers/uhci.c, /kernel/drivers/pciListarDispositivos.c, /kernel/drivers/pci.c, include/kernel/drivers/uhci.h, /include/kernel/drivers/pci_def.h, /include/kernel/drivers/pciListarDispositivos.h, /include/kernel/drivers/pci.h

1.3.3 Nuevas Funciones

Funciones integradas del trabajo práctico: Placa de Red.

Funciones integradas del trabajo práctico: USB.

Funciones a utilizar por otros drivers para comunicarse con el driver PCI

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
vFnInicializarControladoraPCI	Inicia la controladora PCI.	Ninguno (void)	Nada (void)	kernel/main.c
u16FnDetectaBusPCI	Detecta si existe el bus PCI	Ninguno (void)	Entero de 16 bits, la lectura del puerto PCI_DATA_REG	kernel/drivers/pci.c
vFnRecorrerBusPCI	Se encarga de hacer el polling inicial. Iterar por los 256 posibles buses PCI y a su vez por los posibles 32 dispositivos conectados a cada bus. Por ultimo, si es multifunción recorrer las posibles 8 funciones. También se encarga configurar los dispositivos.	Ninguno (void)	Nada (void)	kernel/drivers/pci.c
u32FnLeerConfigPCI	Lee registros de 32 bits de la estructura PCI	- pstuDispositivoPCIparam, puntero a una estructura de dispositivo PCI, sobre el cual quiero leer. - u32Desplazamiento: contiene que registro/s, dentro de la estructura de información PCI, quiero acceder.	Entero de 32 bits, registro de configuracion PCI leido.	kernel/drivers/pci.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
vFnEscribirConfigPCI	Escribe un registro de 32 bits (double word) de la estructura PCI, de un dispositivo en particular	- pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI, sobre el cual quiero escribir. - u32Desplazamiento: contiene que registro/s, dentro de la estructura de informacion PCI, quiero acceder. - u32Valor: el valor que voy a escribir en el registro del dispositivo.	Nada (void)	kernel/drivers/pci.c
u32FnPCIArmarDireccion	Armar un registro de 32 bits, denominado CONFIG_ADDRESS	- stuDispositivoPCIparam: Dispositivo PCI sobre el cual determinar la direccion. - iRegistro: Registro del dispositivo que deseamos acceder.	Entero de 32 bits, Direccion de dispositivo PCI.	kernel/drivers/pci.c
u32FnPciLeerDoubleWord	Leer información provista por un PCI específico.	u32Direccion, Direccion del PCI.	Entero de 32 bits, Dato leido en PCI_DATA_REG	kernel/drivers/pci.c
vFnPciEscribirDoubleWord.	Escribir datos en un PCI específico.	-u32Direccion, Direccion del PCI. - u32Datos: Datos a enviarle (32 bits).	Nada (void)	kernel/drivers/pci.c
vFnOutPortDoubleWord	Escribir un double word (32 bits) en un puerto	-u32Puerto: Numero del puerto a escribir. - u32Valor: Valor a escribir (32 bits).	Nada (void)	kernel/drivers/pci.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
u32FnInPortDoubleWord	Leer un double word (32 bits) de un puerto.	u32Puerto: Numero del puerto a escribir	Entero 32 bits, u32Valor: Dato leido	kernel/drivers/pci.c
vFnLeerstuDispositivoPCI	Leer y mostrar por pantalla una estructura de dispositivo PCI.	u32Puerto: Numero del puerto a escribir	Nada (void)	kernel/drivers/pci.c
vFnHabilitarDispositivoPCI.	Inicializar los recursos PCI de un dispositivo, para ello determina cuales son sus espacios de entrada/salida, de memoria, si los tuviera, y la línea de interrupción.	pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI	Nada (void)	kernel/drivers/pci.c
FnConfigurarDispositivoPCI	Configurar un dispositivo PCI detectado.	pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI	Nada (void)	kernel/drivers/pci.c
vFnAgregarDispositivoEnVector	Agrega la estructura del dispositivo en un vector de estructuras.	pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI.	Nada(void)	kernel/drivers/pci.c
vFnVerDispositivosEnVector	Recorre el vector de estructura para mostrar la información de los dispositivos.	Ninguno (void)	Nada(void)	kernel/drivers/pci.c
vFnIniciarDispositivoPCI	Es el encargado de buscar el driver específico de un dispositivo en particular.	pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI.	Nada(void)	kernel/drivers/pci.c
u8FnBuscarDriverEspecificoDispositivo	Se encarga de buscar un driver para el dispositivo por Id fabricante y Id de dispositivo o por clase y subclase.	pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI.	Retorna un u8, que es un flag, fue exitosa la búsqueda del driver específico	kernel/drivers/pci.c
u32FnAgregarControladoraUSB	Llama a la función encargada de instalar a la controladora USB	pstuDispositivoPCIparam: puntero a una estructura de dispositivo PCI.	Retorna un u32 para indicar se llamo a la función.	kernel/drivers/pci.c
u8 stau8InPortB	Lee un byte (8bit) de un puerto.	u32Puerto: Numero del puerto a leer	Dato leido (8 bits).	kernel/drivers/pci.h
stau16InPortWord	Lee un word (16bit - 2Byte) de un puerto.	u32Puerto: Numero del puerto a leer	Dato leido (16 bits).	kernel/drivers/pci.h
stau32InPortLong	Lee un long (32bit - 4Byte) de un puerto.	u32Puerto: Numero del puerto a leer.	Dato leido (32 bits).	kernel/drivers/pci.h

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
stavOutPortB	Escribe un byte en un puerto.	u32Puerto: Numero del puerto a escribir.	u8Valor: Valor a escribir (8 bits).	kernel/drivers/pci.h
stavOutPortWord	Escribe un word en un puerto.	u32Puerto: Numero del puerto a escribir.	u16Valor: Valor a escribir (16 bits).	kernel/drivers/pci.h
stavOutPortLong	Escribe un long en un puerto.	u32Puerto: Numero del puerto a escribir.	u32Valor: Valor a escribir (32 bits).	kernel/drivers/pci.h
stau32FnPciLeer	Lee la información provista por un PCI específico.	32Direccion: Dirección del PCI.	Dato leído (32 bits).	kernel/drivers/pci.h
stau16FnPciLeerWord	Lee la información provista por un PCI específico.	u32Direccion: Direccion del PCI.	Dato leído (16 bits).	kernel/drivers/pci.h
stau8FnPciLeerByte	Lee la información provista por un PCI específico.	u32Direccion: Direccion del PCI.	Dato leído (8 bits).	kernel/drivers/pci.h
stavFnPciEscribir	Escribe datos en un PCI específico.	- u32Direccion: Direccion del PCI. - u32Datos: Datos a enviarle (32 bits)	Nada(void)	kernel/drivers/pci.h
stavFnPciEscribirWord	Escribe datos en un PCI específico.	- u32Direccion: Direccion del PCI. - u32Datos: Datos a enviarle (16 bits).	Nada(void)	kernel/drivers/pci.h
stavFnPciEscribirByte	Escribe datos en un PCI específico.	- u32Direccion: Direccion del PCI. - u32Datos: Datos a enviarle (8 bits).	Nada(void)	kernel/drivers/pci.h
pvFnPciDmaObtenerVirtual	Obtiene la direccion virtual de una direccion.	pvDireccion: Direccion sobre la cual trabajar.	Direccion virtual	kernel/drivers/pci.c
u32FnPciDmaObtenerFisica	Obtiene la direccion fisica de una direccion.	pvDireccion: Direccion sobre la cual trabajar.	Direccion fisica. (Entero 32bits)	kernel/drivers/pci.c
u32 u32ObtenerDispositivoPCI	Busca el numero de dispositivo en el vector de dispositivos en memoria	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Numero de dispositivo	kernel/drivers/pci.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
u16FnObtenerIdFabricante	Obtiene el ID del fabricante en el vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	ID del fabricante	kernel/drivers/pci.c
u16FnObtenerIdDispositivo	Obtiene el ID del dispositivo en el vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	ID del dispositivo	kernel/drivers/pci.c
u16FnObtenerEstado	Obtiene el estado actual del dispositivo	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Estado del dispositivo	kernel/drivers/pci.c
u16FnObtenerComando	Obtiene el comando actual del dispositivo	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Comando del dispositivo	kernel/drivers/pci.c
vFnEscribirComando	Escribe un comando enviado como parámetro en el dispositivo indicado	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo u16Comando: Comando a setear	Nada(void)	kernel/drivers/pci.c
u16FnObtenerClase	Obtiene la clase del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Clase del dispositivo	kernel/drivers/pci.c
u16FnObtenerSubClase	Obtiene la subclase del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Subclase del dispositivo	kernel/drivers/pci.c
u16FnObtenerInterfaz	Obtiene la interfaz del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Interfaz del dispositivo	kernel/drivers/pci.c
u16FnObtenerIdRevision	Obtiene la revisión del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Revisión del dispositivo	kernel/drivers/pci.c
u16FnObtenerTipoCabecera	Obtiene el tipo de cabecera del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Tipo de cabecera del dispositivo	kernel/drivers/pci.c
u16FnObtenerTemporizacionLatencia	Obtiene la latencia del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Latencia del dispositivo	kernel/drivers/pci.c
u16FnObtenerTamanioCache	Obtiene el tamaño de la memoria cache del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Tamaño de la memoria cache del dispostivo	kernel/drivers/pci.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
u16FnObtenerIdSubsistema	Obtiene el ID del subsistema del dispositivo del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Id del subsitema del dispositivo	kernel/drivers/pci.c
u16FnObteneruIdFabricanteSubsistema	Obtiene el ID del fabricante del subsistema del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Id del fabricante del subsitema	kernel/drivers/pci.c
u8FnObtenerPinInterrupcion	Obtiene el pin de interrupción del vector de dispositivos	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Pin de interrupción	kernel/drivers/pci.c
u8FnObtenerLineaInterrupcion	Obtiene la línea de interrupción del dispositivo	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Línea de interrupción	kernel/drivers/pci.c
vFnEscribirLineaInterrupcion	Escribe una línea de interrupción enviada como parámetro en el dispositivo indicado	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo u8LineaInterrupcion: Linea de interrupcion a setear	Nada(void)	kernel/drivers/pci.c
u32FnObtenerDireccionBar	Obtiene la dirección del BAR enviado como parámetro del dispositivo indicado	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo u32Bar: Numero de Bar	Dirección del Bar	kernel/drivers/pci.c
u32FnObtenerTamanioBar	Obtiene el tamaño del BAR enviado como parámetro del dispositivo indicado	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo u32Bar: Numero de Bar	Tamaño del Bar	kernel/drivers/pci.c
vFnFuncionPrueba	Utilizada para probar las funciones generales	u16IdFabricante: Id del fabricante u16IdDispositivo Id del dispositivo	Nada(void)	kernel/drivers/pci.c
vFnListarDispositivos	Imprime por pantalla la lista de dispositivos PCI	Ninguno (void)	Nada (void)	kernel/drivers/pciListarDispositivos.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
vFnObtenerFabricante	En base al parámetro ,u16IdFabricante, copia el nombre del fabricante y del dispositivo en el parametro stFabricante. Verifica que exista fabricante.	-u16IdFabricante: contiene el Id del fabicante. -stFabricante recibe el nombre del fabricante y del dispositivo.	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDispositivo	En base a los parámetros ,u16IdFabricante y u16IdDispositivo, copia el nombre del fabricante y del dispositivo en el parametro stFabricante. Verifica que exista dispositivo.	-u16IdFabricante: contiene el Id del fabicante. -stFabricante recibe el nombre del fabricante y del dispositivo. -u16IdDispositivo: contiene el Id del dispositivo	Nada (void)	kernel/drivers/pciListarDispositivos.c
vFnListarDispositivosEspecifico	Imprime por pantalla la descripcion completa de un dipositivo enviado como parámetro.	iN: Numero de dispositivo a listar	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivo	Con los parametros u16Clase y u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion	-u16Clase: parametro que contiene la clase del dispositivo -u16SubClase: parametro que contiene la subclase del dispositivo -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoRed	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
vFnObtenerDescripcionDispositivoMass	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo.	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoDisplay	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoMultimedia	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoMemory	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoBridge	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
vFnObtenerDescripcionDispositivoCommunication	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoSystem	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoInput	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoDock	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoProcessor	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo.	Nada (void)	/kernel/drivers/pciListarDispositivos.c

Nombre Función	Objetivo	Parámetros	Retorno	Ubicación
vFnObtenerDescripcionDispositivoSerialBus	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoWireless	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoSatcom	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoCrypto	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c
vFnObtenerDescripcionDispositivoDasp	En base al parametro u16SubClase copia la descripcion del dispositivo en el parametro stDescripcion.	-u16SubClase, parametro que contiene la sublclase a la cual pertenece el dispositivo. -stDescripcion parametro que recibe la descripcion del dispositivo	Nada (void)	/kernel/drivers/pciListarDispositivos.c

1.3.4 Casos De Prueba.

1.3.4.1. Comando listar dispositivos.

- **Entrada**: lspci
- **Proceso**: lista los dispositivos con el formato:

Bus: Dispositivo: Función: Descripción del fabricante: Revisión del dispositivo

- **Salida:**

```
Bienvenidos al SODIUM...
Cmd>lspci

Lista de dispositivos PCI
-------------------------
Bus:Dis:Fun  Descripcion Fabricante Dispositivo Rev

 0:0:0 Bridge Host: Intel Corporation 82440LX/EX [PCI & Memory] (Rev: 0) 8086:12
37:6:0
 0:1:0 Bridge ISA: Intel Corporation 82371SB [PIIX3 PCI-to-ISA Bridge (Triton II
)] (Rev: 0) 8086:7000:6:1
 0:1:1 Almacenamiento Masivo IDE: Intel Corporation 82371SB [PIIX3 IDE Interface
 (Triton II)] (Rev: 0) 8086:7010:1:1
 0:1:2 Serial Bus USB: Intel Corporation 82371SB [PIIX3 USB Host Controller (Tri
ton II)] (Rev: 1) 8086:7020:C:3
 0:1:3 Bridge MISC: Intel Corporation 82371AB/EB/MB [PIIX4/4E/4M Power Managemen
t Controller] (Rev: 3) 8086:7113:6:80
-------------------------

Cmd>
```

1.3.4.2. Comando listar dispositivos de forma detallada

- **Entrada**: lspci -v
- **Proceso**: lista los dispositivos con el formato:

Bus: Dispositivo: Función:
Descripción del dispositivo: Clase: Subclase:
Descripción del fabricante.

- **Salida:**

```
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>lspci -v


DESCRIPCION DE DISPOSITIVO: 0
Bus: 0  Dispositivo: 0  Funcion: 0
Descripcion: Bridge Host Clase: 6 Sublcase: 0
Fabricante: Intel Corporation [8086]

Presione Cualquier tecla para continuar, o "s" para salir...

DESCRIPCION DE DISPOSITIVO: 1
Bus: 0  Dispositivo: 1  Funcion: 0
Descripcion: Bridge ISA Clase: 6 Sublcase: 1
Fabricante: Intel Corporation [8086]

Presione Cualquier tecla para continuar, o "s" para salir...
```

1.3.4.3. Comando listar dispositivo específico de forma detallada

- **Entrada**: lspci [numero de dispositivo]
- **Proceso**: lista el dispositivo indicado con el formato:

  Bus: Dispositivo: Función:
  Descripción del dispositivo: Clase: Subclase:
  Descripción del fabricante.

- **Salida:**

```
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>
Cmd>lspci 1

DESCRIPCION DE DISPOSITIVO: 1
Bus: 0  Dispositivo: 1  Funcion: 0
Descripcion: Bridge ISA Clase: 6 Sublcase: 1
Fabricante: Intel Corporation [8086]

Cmd>_
```

1.4 Archivos Fuentes

1.4.1 Comentarios por funciones y archivos

1.4.2 Comentarios en las funciones

1.4.3 Nomenclatura Sodium

2. Conclusiones

2.1. Problema al desarrollar

2.1.1 Problema con archivos USB

Al abordar la codificación de la controladora PCI y sus archivos: pci.c y pci.h se encontraron "hardcodeadas" muchas funciones y variables relativas al driver USB.
El objetivo, para evitar errores difíciles de corregir, fue comenzar desde cero con el desarrollo del driver, por lo tanto se eliminaron tanto las referencias como los archivos del driver USB de los directorios de SODIUM. Luego de estas actividades comenzamos comenzar con la programación del driver.

2.1.2 Problema con el SVN

Tuvimos algunos problemas con la utilización del SVN para codificar. Al copiar los archivos de una carpeta a otra, no copiaba las carpetas /usr/ y /solo/. La solución fue una copia de estos archivos de forma recursiva, no pudimos encontrar el motivo que provocaba que no se copien estas carpetas.

2.1.3 Problema con la memoria de SODIUM

Una vez integrados los archivos de USB, nos encontramos con el problema que SODIUM compilaba pero no iniciaba. El problema radicaba en la cantidad de memoria que no era suficiente para poder integrar estos archivos. La solución fue quitar el archivo shoutdown.bin para liberar memoria.

2.2. Conclusión

La versión de SODIUM sobre la que se comenzó el desarrollo contenía un driver PCI limitado y muy dependiente del driver para dispositivos USB. La primera tarea fue eliminar esas dependencias para comenzar a desarrollar un driver PCI genérico. A continuación, identificamos los puertos de comunicación utilizados para intercambiar información con la CPU y recuperar datos de la memoria principal. Luego se comprendieron los diferentes registros que identifican y permiten gestionar un dispositivo PCI. El driver PCI desarrollado permite identificar a cada uno de los dispositivos que se pueden conectar a la computadora a través del bus PCI y puede

actuar como interfaz para los drivers específicos de los mismos. Con esto se ha ampliado la capacidad de comunicación de SODIUM con el mundo exterior.

3. Evolución de TP (Mejoras)

En una próxima etapa se puede hacer que el driver brinde la funcionalidad plug & play, por el momento, si bien el driver hacer el recorrido de los buses cada cierto intervalo de tiempo no llama al driver específico del dispositivo encontrado, sólo se limita a identificarlo y asignarle los recursos que éste requiera.

Mejoras ha realizar:

1. El poolling de dispositivos, que en la actualidad se realizar en MODO PROTEGIDO, se debería realizarse, como en se hace en la actualidad en los sistemas operativos, en MODO REAL. Esto será posible cuando se mejore la cantidad de memoria que utiliza SODIUM en MODO REAL.
2. Por ser SODIUM un sistema didáctico, se podría dotarlo de la posibilidad de realizar distintos tipos de de polling y elegir cual usar al inicial el S.O:
   - Polling de fuerza bruta: recorrer los 256 buses, 32 dispositivos y 8 funciones buscando todos los dispositivos, existan o no.
   - Polling recursivo: recorrer el primer bus, chequear si es un puente, si lo es, extraer el bus secundario y recorrerlo.
   - Polling recursivo con configuración de bus: similar al polling recursivo con la posibilidad de configurar el bus secundario.

   Más información: http://wiki.osdev.org/PCI#Base_Address_Registers
3. Investigar el uso de CardBus que actualmente es el resultado evolutivo de lo que originalmente fuera el PCMCIA. Es necesario para completar la inicialización de estos dispositivos conectados al bus PCI.
4. Cambiar el vector utilizado para almacenar las estructuras de los dispositivos por el uso de punteros que es más eficiente en el manejo de memoria.
5. Incorporar una mayor cantidad de IDs de Fabricantes y IDs de Dispositivos para reconocer los dispositivos PCI. La información puede ser extraída de la página: http://www.pcidatabase.com/ que permite descargar un Header.C o generar un archivo separado por comas.
6. En una próxima etapa se puede hacer que el driver brinde la funcionalidad plug & play, ya que en esta versión hace el recorrido de los buses cuando el sistema operativo inicia. Tampoco llama al driver específico del dispositivo encontrado (solo al de USB, sólo se limita a identificarlo y asignarle los recursos que éste requiera.

4. Apéndice

4.1 Desarrollo de drivers específicos

La información de configuración de cada dispositivo conectado al bus PCI podrá ser obtenida utilizando las funciones resaltadas en amarillo en la tabla de funciones. A través de éstas cualquier driver podrá identificar el dispositivo a administrar, obtener su información de estado, conocer sus espacios de memoria y entrada/ salida, obtener su

línea de interrupción y conocer el tamaño de sus BARs. El driver a instalar deberá modificar el archivo PCI.c para incorporar el llamado a si mismo durante el inicio de cada dispositivo.

4.1.1 Consideraciones especiales para el Driver USB

Debido a que el bus PCI no brinda la funcionalidad "hot plug-in" el pooling de dispositivos se realiza al iniciar el sistema operativo. En el caso particular de los dispositivos USB, el driver PCI va a administrar los Hubs USB que se conecten al mismo. Por lo tanto, será responsabilidad del driver USB la detección, configuración y administración de cualquier dispositivo USB que se conecte a al hub USB en cuestión.

4.2 Desarrollo de drivers específicos

En la siguiente figura observamos la relación entre el driver PCI y los drivers de los dispositivos. Cada driver particular administra y gestiona los dispositivos mientras que el driver PCI recibe las peticiones de cada driver y arbitra el uso de los recursos compartidos por ejemplo la CPU.