Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

УТВЕРЖДАЮ
Директор ИКНК
_____ Д.П. Зегжда
«17» июня 2024 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

«Программирование драйверов периферийных устройств»

Разработчик	Высшая школа программной инженерии
Направление (специальность) подготовки	09.03.04 Программная инженерия
Наименование ООП	09.03.04_01 Технология разработки и сопровождения качественного программного продукта
Квалификация (степень) выпускника	**бакалавр**
Образовательный стандарт	**СУОС**
Форма обучения	**Очная**

СОГЛАСОВАНО
Руководитель ОП
_____ А.В. Петров
«21» мая 2024 г.

Соответствует СУОС
Утверждена протоколом заседания высшей школы "ВШПИ"
от «21» мая 2024 г. № 1

РПД разработал:
Специалист по учебно-методической работе 1 категории Т.А. Вишневская

1. Цели и планируемые результаты изучения дисциплины

Цели освоения дисциплины

Целью дисциплины является формирование специалистов, умеющих обоснованно и результативно разрабатывать программное обеспечение использовать современную вычислительную технику, владеющих технологиями разработки программ, функционирующих в режиме ядра ОС, способных освоить существующие и появляющиеся средства и технологии.

Результаты обучения выпускника

Код	Результат обучения (компетенция) выпускника ООП
ПК-10	**Способен применять в практической деятельности основные концепции разработки программно-аппаратных комплексов**
ИД-2 ПК-10	Разрабатывает драйверы устройств для программно-аппаратных комплексов с использованием современных технологий и программных средств

Планируемые результаты изучения дисциплины

знания:

- Знает основные понятия, архитектуры и принципы построения ЭВМ и системного программного обеспечения

умения:

- Умеет проектировать и реализовывать компоненты системного программного обеспечения

навыки:

- Владеет современными средствами разработки, тестирования и отладки программного обеспечения

2. Место дисциплины в структуре ООП

В учебном плане дисциплина «Программирование драйверов периферийных устройств» относится к модулю «Программно-аппаратные комплексы. Электив».

Изучение дисциплины базируется на результатах освоения следующих дисциплин:

- Архитектура программных систем
- Операционные системы

• Системное программное обеспечение GNU/Linux

3. Распределение трудоёмкости освоения дисциплины по видам учебной работы и формы текущего контроля и промежуточной аттестации

3.1. Виды учебной работы

Виды учебной работы	Трудоемкость по семестрам
	Очная форма
Лекционные занятия	30
Практические занятия	4
Самостоятельная работа	37
Часы на контроль	16
Промежуточная аттестация (экзамен)	11
Курсовое проектирование	10
Общая трудоемкость освоения дисциплины	108, ач
	3, зет

3.2. Формы текущего контроля и промежуточной аттестации

Формы текущего контроля и промежуточной аттестации	Количество по семестрам
	Очная форма
Текущий контроль	
Курсовые проекты, шт.	1
Промежуточная аттестация	
Экзамены, шт.	1

4. Содержание и результаты обучения

4.1 Разделы дисциплины и виды учебной работы

№ раздела	Разделы дисциплины, мероприятия текущего контроля	Очная форма		
		Лек, ач	Пр, ач	СР, ач
1.	Архитектура и технические средства обмена данными с периферийными устройствами.			

1.1.	Системная организация вычислительных сисем.	2	0	0
1.2.	Архитектура ввода-вывода в системах на базе процессоров семейства x86.	1	0	2
1.3.	Асинхронные прерывания.	1	1	3
2.	Архитектура системного программного обеспечения ввода-вывода в современных операционных системах.			
2.1.	Архитектура операционных систем с точки зрения организации обмена с периферийными устройствами.	2	0	0
2.2.	Компоненты операционных систем для обеспечения ввода-вывода.	2	0	0
2.3.	Интерфейсы ввода-вывода для программ пользовательского режима.	2	0	2
2.4.	Пользовательские операции ввода-вывода с точки зрения разработчиков драйвера.	2	0	3
3.	Технологии создания модулей расширения ядра операционной системы.			
3.1.	Компоновка и связывание компонентов программ.	2	1	2
3.2.	Средства разработки модулей ядра, драйверов режима ядра и драйверов в пространстве пользовательского процесса.	2	1	3
3.3.	Вспомогательные средства поддержки разработки драйверов.	2	0	2
3.4.	Создание драйверов в пространстве ядра.	2	1	3
4.	Программные средства поддержки функционирования драйверов устройств.			
4.1.	Жизненный цикл драйвера с точки зрения программиста.	2	0	1
4.2.	Программная поддержка времени выпонения.	2	0	2
4.3.	Специальные приемы программирования драйверов.	2	0	3
4.4.	Специальные приемы работы с аппаратурой из кода драйвера.	2	0	3
4.5.	Программирование периферийных устройства из пользовательской программы.	2	0	2
Итого по видам учебной работы:		30	4	37
Экзамены, ач				16
Часы на контроль, ач				16
Курсовое проектирование		10		
Промежуточная аттестация (экзамен)		11		

Общая трудоёмкость освоения: ач / зет	108 / 3

4.2. Содержание разделов и результаты изучения дисциплины

Раздел дисциплины	Содержание
1. Архитектура и технические средства обмена данными с периферийными устройствами.	
1.1. Системная организация вычислительных сисем.	Исторический обзор. Современная ситуация и тенденции. Периферийные устройства. Иерархическая организация. Системные и периферийные магистрали. Модель программиста: инструкции для обеспечения ввода-вывода, канальная архитектура, обмен с памятью под управлением устройства. Непосредственный обмен и опосредованный обмен через специализированный контроллер интерфейса.
1.2. Архитектура ввода-вывода в системах на базе процессоров семейства x86.	Типовая система на базе IA32 процессора. Шины и мосты. Периферийные магистрали и их интерфейсы. Шины PCI, PCIe и их разновидности. Возможности, типовые операции. Обнаружение и настройка устройств. Технология Plug and Play. Интерфейс USB: архитектура, компоненты, версии, типовые Host контроллеры. Команды ввода-вывода и ввод-вывод, отображенный на память в IA32. Архитектура памяти с точки зрения программирования обмена с периферийными устройствами. Программирование в инструкциях и на языке C. Потребность в уровне абстракции аппаратуры.
1.3. Асинхронные прерывания.	Потребность в асинхронных прерываниях. Уровневые и фронтовые сигналы. Контроллеры прерываний. Архитектура APIC (advanced programmable interrupt controller). Локальный контроллер прерываний на примере Local APIC IA32. Контроллер прерываний ввода-вывода на примере IOAPIC. Прерывания, основанные на передаче сообщений MSI (message signalling interrupt). Поддержка прерываний в архитектуре IA32, таблица прерываний, шлюзы прерываний и вектора. Прерывания в мультипроцессорных системах. Рассмотрение системы прерываний в типовой системе с IA32 процессором.
2. Архитектура системного программного обеспечения ввода-вывода в современных операционных системах.	

2.1. Архитектура операционных систем с точки зрения организации обмена с периферийными устройствами.	Назначение операционных систем. Типы операционных систем. Микроядерная и монолитная архитектура. Особенности реализации обслуживания ввода-вывода в микроядерной и монолитной архитектуре. Операционные системы, реализующие концепцию выполнения кода в режиме ядра и пользовательских процессов. Аппаратурная поддержка режимов работы процессора, уровни привилегий, защита памяти и ресурсов ввода-вывода. Механизм системных вызовов. Примеры реализаций механизма системных вызовов в IA32 архитектуре.
2.2. Компоненты операционных систем для обеспечения ввода-вывода.	Операционные системы с возможностью расширения. Операционные системы с открытым и закрытым исходным кодом. Связывание компонентов. Бинарные программные интерфейсы: наличие или отсутствие. Модульная архитектура. Драйверы устройств как модули расширения ядра. Рассмотрение распространенных операционных систем как расширяемых вычислительных систем. Примеры. ОС на базе ядра Linux. Варианты оформления модулей. Системы с архитектурой Windows NT.
2.3. Интерфейсы ввода-вывода для программ пользовательского режима.	Взаимодействие программ пользовательского режима и модулей ядра. Сценарии выполнения обмена. Синхронные и перекрывающиеся операции. Примеры. Операции ввода-вывода в ОС архитектуры Windows NT. Менеджер ввода-вывода Windows NT.

2.4. Пользовательские операции ввода-вывода с точки зрения разработчиков драйвера.	Обслуживание запросов ввода-вывода в модуле расширения ядра. Обслуживающие функции, привязка обслуживающих функций. Организация обслуживания на основе файловых операций. Типовые абстракции. Файловые объекты. Объекты драйвера и объекты устроств. Назначение. Обслуживание множества экземпляров устройств в драйверах. Обслуживание запросов ввода-вывода в ОС на основе Windows NT. Жизненный цикл запросов ввода-вывода. Синхронное и отложенное завершение. Обмен данными с клиентом драйвера: пользовательский буфер, буферизированный ядром и прямой доступ к пользовательскому буферу. Отображение данных в пространстве ядра. Виртуальные и физические адреса буфера. Фрагментация. Список дескрипторов памяти буфера. Использование для прямого доступа к пользовательскому буферу кодом режимая ядра и при программировании периферийных устройств, использующих прямой доступ к памяти. Драйверная архитектура ядра Windows NT. Организация драйверов: монолитные и многоуровневые драйверы. Драйверы классов и минидрайверы.
3. Технологии создания модулей расширения ядра операционной системы.	

3.1. Компоновка и связывание компонентов программ.	Понятие о связывании программных компонентов. Статичное и динамическое связывание, динамическая загрузка. Варианты компоновки на примере ядра Linux: статично построенное ядро и компиляция в дереве исходных текстов для ядра без поддержки загружаемых модулей, ядро с поддержкой загружаемых модулей и компиляция в и вне дерева исходных кодов с динамической загрузкой модуля. Модули на основе ELF формата: информация для связывания. Связывание загружаемых модулей ядра в ОС архитектуры Windows NT: модули на основе формата PE. Импорт адресов. Функции загрузчика. Позиционно-завивисимый и позиционно-независимый код, роль таблицы перемещаемых символов. Рассмотрение примера загрузки модуля формата PE в пространстве пользователя и пространстве ядра при помощи отладчика. Низкоуровневые и высокоуровневые средсва компиляции модулей. Утилиты анализа заголовков модулей, анализ дампов и дизассемблирование на примере PE формата для IA32 архитектуры.

3.2. Средства разработки модулей ядра, драйверов режима ядра и драйверов в пространстве пользовательского процесса.	Требования к программным средствам создания модулей ядра. Используемые языки программирования. Архитектурно-зависимые и архитектурно-независимые средства. Использование языка С, ассемблера, втроенного ассемблера. Возможности использования объекто-ориентированных языков и технологий. Библиотеки и средства поддержки создания драйверов. Обзор программных средств создания модулей расширения ядра и драйверов для ОС архитектуры Windows NT. Использование свободно распространяемых и коммерческих программных средств, достоинства, недостатки, ограничения. Программные средства от разработчиков ОС архитектуры Windows NT: DDK (Driver Develipment Kit - набор для разработки драйверов), KMDF (Kernel Mode Driver Framework - библиотека и каркас разработки драйверов режима ядра), UMDF (User Mode Driver Framework - библиотека и карас разработки драйверов в пространстве пользователя). Возможности, абстракции, подходы к разработке. Основные сценарии разработки. Обзор основных средств каркаса для драйверов различного назначения и по типам устройств. Краткий обзор других продуктов. Возможность использования альтернативных среств разработки на базе свободно распространяемого ПО на примере GNU GCC. Возможности встраивания рассмотренных программных стредств в интегрированные среды разработки.

3.3. Вспомогательные средства поддержки разработки драйверов.	Программы анализа бинарных модулей и их зависимостей. Программы анализа и редактирования конфигурации системы (системного реестра, установочных файлов, управление системными службами). Программы отображения диагностической информации времени выполнения: просмотр сообщений протокола работы, просмотр списка объектов ядра (файловых объектов, символьных ссылок, выделения памяти ядра), программы мониторинга активности различных подсистем (ввода-вывода, управления конфигурацией), программы мониторинга устройств (наблюдение за USB шиной). Приемы отладки драйверов: отладочный вывод, протоколирование работы, проверка условий. Использование специальных отладочных вариантов ядра. Использование условной компиляции. Сведения о возможностях активной отладки кода режима ядра: технологии отладки, локальная и удаленная отладка. Обзор специализированных программ отладчиков. Рассмотрение примеров отладки простых модулей ядра. Программы создания тестового окружения: нормальное и стрессовое тестирование. Типовые ошибки: утечки ресурсов, неверный контекст выполнения определенных операций, блокировки, состязательные состояния, неверный контекст обращения к вытесняемой странице памяти, ошибки прямого доступа к пользовательскому буферу по указателю. Неверные предположения о разрядности арифметических и указательных типов на различных платформах. Неверное предположение о размещении данных меньшей разрядности в многоразрядных словах с точки зрения процессора и устройства. Приемы выявления ошибок. Программы анализа исходных текстов драйверов. Поиск шаблонов ошибок.

3.4. Создание драйверов в пространстве ядра.	Этапы разработки. Выбор типа драйвера: модуль ядра или пользовательский процесс. Выбор архитектуры драйвера: монолитный или многоуровневый. Анализ интегрирования драйвера в систему. Необходимые сведения об управляемом устройстве: интерфейсы, подключение, обнаружение, обмен данными. Выбор средств разработки. Пример создания простого монолитного драйвера, обслуживающего запросы ввода-вывода для устройства без асинхронных прерываний. Установка и удаление драйвера. Подготовка к отладке. Символьная информация. Тестирование. Пример сессии отладки и протоколирования работы.
4. Программные средства поддержки функционирования драйверов устройств.	
4.1. Жизненный цикл драйвера с точки зрения программиста.	Драйверы как сервисы. Основные положения. Инициализация. Регистрация обслуживающих функций. Выполнение операций. Контектсты выполнения кода драйвера. Контекст клиентского процесса. Контекст кода ядра. Потоки режима ядра. Таймеры. Выполнение кода драйвера в контексте асинхронных прерываний. Уровни выполнения: абстракции уровненй выполнения, принятые в драйверной архитектуре ядра ОС Windows NT. Уровни прерываний устройств. Уровень отложенной обработки. Назначение и применение. Ограничения набора допустимых операций для различных уровней выполнения.

4.2. Программная поддержка времени выпонения.	Основные виды поддерживающих функций. Отличия и ограничения по сравнению с программами пользовательсого режима. Поддержка времени выпонения в ядре ОС Windows NT. Поддержка диагностики и отладки. Поддержка регистрации. Поддержка управления памятью. Поддержка работы с запросами ввода-вывода. Поддержка работы с аппаратурой: уровень аппаратурной абстракции, прерывания, поддержка устройств, осуществляющих прямой доступ к памяти или обслуживаемых контроллерами прямого доступа. Поддержка программных потоков и синхронизации. Поддерржка обнаружения и работоы с самонастраиваемыми устройствами (поддержка технологии Plug and Play). Краткие сведения о поддержке драйверов специального назаначения: графика, сетевые устройства, файловые системы.
4.3. Специальные приемы программирования драйверов.	Отложенное выполнение запросов ввода-вывода. Организация очередей запросов. Самостоятельная огранизация очередей и поддержка ядра для организации очередей. Использование асинхронных прерываний, таймеров и программных потоков в коде драйвера. Синхронизация доступа к разделяемым ресурсам из участков кода драйвера, выполняемых в различных контекстах и с различными уровнями выполнения. Специальные методы синхронизации, применяемые при программировании кода режима ядра. Поддержка (объекты и функции) различных методов синхронизации для разработчиков драйверов в ядре ОС Windows NT: события, семафоры, мьютексы. Специальные виды синхронизации в ядре. Спин-блокировки. Оптимизация монопольного и разделяемого по чтению доступа к ресурсам. Атомарные арифметические операции. Поддержка KMDF по организации очередей запросов. Пример драйвера, использующего очереди запросов, асинхронные прерывания и отложенную обработку для выполнения запроса ввода-вывода.

4.4. Специальные приемы работы с аппаратурой из кода драйвера.	Plug and Play устройства в драйверной модели Windows NT. Каркас драйвера для Plug and Play устройства. Добавление и удаление устройства. Ресурсы ввода-вывода. Подключение объекта прерывания. Использование функций библиотеки уровня аппаратурной абстракции для платформенно-независимого доступа к ресурсам устройства ввода-вывода. Использование прямого доступа к памяти платформенно-независимым образом. Поддержка управления питанием. Установка драйверов, поддерживающих автоматический запуск при подключении устройства.
4.5. Программирование периферийных устройства из пользовательской программы.	Основные тенденции в развитии технологий создания программого обеспечения обслуживания устройств ввода-вывода. Драйверы в пространстве пользовательского процесса. Достоинства и недостатки. Требования к системному программному обеспечению и аппатаруре. Каркасы для драйверов пользовательского режима для некоторых видов устройств. Возможности и ограничения. Пример: каркас драйвера пользовательского режима для USB устройства в UMDF. Пример: программирование USB устройства при помощи библиотеки libusb.

5. Образовательные технологии

1. При изучении дисциплины используются традиционные образовательные технологии: лекционные и правктические занятия
2. Изучение дисциплины предполагает опережающую самостоятельную работу по изучению технической докоментации на рассматриваемую аппаратуру
3. Изучение дисциплины предполагает самостоятельный поиск дополнительной информации по приемам разработки драйверов устройств и изучение исходных текстов, созданных разработчиками свободного программного обеспечения

6. Лабораторный практикум

Не предусмотрено

7. Практические занятия

№ раздела	Наименование практических занятий (семинаров)	Трудоемкость, ач
		Очная форма
1.	Ввод-вывод в типовой системе на базе IA32 процессора. Создание базовой конфигурации аппаратуры. Создание конфигурации для отладки. Написание стартового кода для каркаса учебно-демонстрационного приложения. Отладка стартового кода на эмуляторе. Создание каркаса приложения на языке С и его отладка. Добавления простейших средств взаимодействия с периферийными устройствами на примере асинхронного последовательного адаптера. Проверка работоспособности. Модификация базового приложения для поддержки работы с асинхронными прерываниями. Инициализация контроллера прерываний. Проверка по диагностическим сообщениям эмулятора.. Построение примера, демонстрирующего работу асинхронного последовательного адаптера в режиме обслуживания прерываний для приема и передачи символов. Отладка и демонстрация работы на эмуляторе. Контроль нагрузки процессора через средства мониторинга.	1

2.	Технология Plug and Play на примере PCI и USB устройств. Поиск и обнаружение устройства. Выделение ресурсов ввода-вывода. Программирование обмена в режиме прямого доступа под правлением устройства на примере контроллера локальной сети. Создание аппаратурной конфигурации эмулятора и подключение виртуальной системы к стеку сетевых протоколов. Создание тестового окружения для наблюдения приема и передачи пакетов. Работа с программами мониторинга сети. Отладка приложения, принимающего и передающего кадры в режиме прямого доступа. Программирование взаимодействия с USB устройствами. Модификация базового каркаса примера с переиспользованием фрагментов кода примера поиска и настройки PCI устройств для обнаружения и настройки UHCI USB контроллера. Создание конфигурации эмулятора с поддержкой UHCI USB контроллера и возможности добавления виртуального последовательного адаптера с USB интерфейсом. Отладка приложения, обнаруживающего подключение USB устройства к порту контроллера UHCI. Добавление к примеру возможностей выбора конфигурации устройства, настройки адаптера и приема потока символов с выдачей их на другой последовательный адаптер (не USB) для демонстрации функционирования.	1
3.	Создание простейшего модуля расширения ядра для ОС архитектуры Windows NT. Ручная установка и удаление конфигурации службы драйвера ядра, запуск и остановка службы. Последовательная отладка посредством протоколирования отладочного вывода в протокол (журнал) этапов жизненного цикла модуля: инициализация, регистрация обслуживающих процедур, выгрузка. Модификация кода модуля для обслуживания запросов ввода-вывода. Написание и запуск тестового приложения пользовательского режима. Обработка клиентских данных при помощи обмена через системный буфер.	1

4.	Создание драйвера режима ядра, использующего отложенную обработку запроса ввода, завершаемую в процедуре отложенного вызова (DPC), планируемую в процедуре обработки прерывания. для асинхронного последовательного адаптера. Отладка и протоколирование работы драйвера. Модификация драйвера с отложенной обработкой запросов для работы с системной очередью запросов. Написание тестового приложения, использующего перекрывающиеся операции ввода-вывода. Отладка кода драйвера. Обработка отмены запросов, находящихся в обработке. Построение приложения при помощи каркаса KMDF с поддержкой очередей запросов. Подготовка данных для автоматической установки драйвера. Подготовка драйвера PLug and Play устройства на примере USB последовательного адаптера. Создание драйвера, передающего и принимающего блоки символов через USB асинхронный адаптер с использованием каркаса USB драйвера режима ядра. Проверка установки драйвера.	1
	Итого часов	4

8. Организация и учебно-методическое обеспечение самостоятельной работы

Самостоятельная работа направлена на закрепление и углубление освоения учебного материала, развитие практических умений. Самостоятельная работа включает следующие виды самостоятельной работы студентов:

- работа с лекционным материалом и учебной литературой;
- самостоятельное изучение дополнительных технологий программирования драйверов, не освещенных в лекционном материале, каркасов, программных средств;
- освоение других аппаратурных и программных платформ и архитектур, например, других операционных систем, другого ряда архитектур систем команд;
- упреждающее изучение выдаваемых материалов по курсу с целью лучшего восприятия демонстрационных примеров
- изучение электронных ресурсов, дающих практические рекомендации в решении различных типовых и иниых проблем при разработке драйверов
- изучение исходных тектстов драйверов и иниых компонентов системного программного обеспечения с открытым исходным кодом

Методом контроля для работы с лекционным материалом и подготовки к экзамену является контроль преподавателем результатов работ студентов при демонстрации иим функционирования исходных и модифицированных примеров на практических занятиях.

Примерное распределение времени самостоятельной работы студентов

Вид самостоятельной работы	Примерная трудоемкость, ач
	Очная форма
Текущая СР	
работа с лекционным материалом, с учебной литературой	4
опережающая самостоятельная работа (изучение нового материала до его изложения на занятиях)	4
самостоятельное изучение разделов дисциплины	6
выполнение домашних заданий, домашних контрольных работ	4
подготовка к лабораторным работам, к практическим и семинарским занятиям	4
подготовка к контрольным работам, коллоквиумам	0
Итого текущей СР:	22
Творческая проблемно-ориентированная СР	
выполнение расчётно-графических работ	0
выполнение курсового проекта или курсовой работы	0
поиск, изучение и презентация информации по заданной проблеме, анализ научных публикаций по заданной теме	3
работа над междисциплинарным проектом	0
исследовательская работа, участие в конференциях, семинарах, олимпиадах	0
анализ данных по заданной теме, выполнение расчётов, составление схем и моделей на основе собранных данных	6
Итого творческой СР:	9
Общая трудоемкость СР:	37

9. Учебно-методическое обеспечение дисциплины

9.1. Адрес сайта курса

https://dl.spbstu.ru/course/view.php?id=1910

9.2. Рекомендуемая литература

Основная литература

№	Автор, название, место издания, издательство, год (годы) издания	Год изд.	Источник
1	Тышкевич А.И., Медведев Б.М. Использование эмулятора qemu для разработки программ управления периферийными устройствами, 2022. URL: https://elib.spbstu.ru/dl/5/tr/2022/tr22-109.pdf	2022	ЭБ СПбПУ

Дополнительная литература

№	Автор, название, место издания, издательство, год (годы) издания	Год изд.	Источник
1	Нортон Д.А. Написание драйверов для Windows: М.: Мир, 1994.	1994	ИБК СПбПУ

Ресурсы Интернета

1. Информация по устройствам, рассматриваемым в этом курсе, а также устройствам других типов.: https://wiki.osdev.org/Main_Page.

9.3. Технические средства обеспечения дисциплины

Для выполнения работ на практических занятиях студенты по своему выбору могут использовать
персональные компьютеры (личные или имеющиеся в учебных классах). Студенты самостоятельно выбирают необходимые им программные средства (раздел 5): свободно используемые или имеющееся в их распоряжении лицензионное программное обеспечение.

Для лекционных и практических занятий имеется набор разработанных иллюстративных примеров, включающих исходный тектст небольшой программы, файлы проекта для самостоятельного построения в среде Eclipse IDE, текстовые файлы с примерами команд создания конфигураций эмулятора QEMU и вспомогательным утилитам.

Также к лекционным ит практическим занятиям предоставляюся:

- Презентация по занятию со снимками экрана сессий отладки и демонстрации примеров
- Описание примера с иллюстрациями, ссылками на техническую документацию и указаниями по внесению изменений примеры для самостоятельной работы

Материалы предоставляются для упреждающего изучения перед практическим занятием для самостоятельной подготовки.

10. Материально-техническое обеспечение дисциплины

Проекционное оборудование в аудитории для лекционных и практических занятий, желательно с высоким разрешением (от 1920x1024) для рассмотрения одновременной работы нескольких программ и многооконных интерфейсов.

Компьютерное оборудование с поддержкой технологии аппаратурного ускорения виртуализации Intel VT-x.

Программное обеспечение

Свободное:

- ОС на основе Linux с исходными текстами
- Eclipse IDE
- GDB
- GCC
- QEMU

Лицензионное:

- ОС Windows (образ для установки, любой от Windows 7)
- Microsoft Visual Studio версий старше 2012 г.

11. Критерии оценивания и оценочные средства

11.1. Критерии оценивания

Для дисциплины «Программирование драйверов периферийных устройств» формой аттестации является экзамен. Дисциплина реализуется с применением системы индивидуальных достижений.

Текущий контроль успеваемости

Максимальное значение персонального суммарного результата обучения (ПСРО) по приведенной шкале - 100 баллов
Максимальное количество баллов приведенной шкалы по результатам прохождения двух точек контроля - 80 баллов.

Подробное описание правил проведения текущего контроля с указанием баллов по каждому контрольному мероприятию и критериев выставления оценки размещается в СДО в навигационном курсе дисциплины.

Промежуточная аттестация по дисциплине

Максимальное количество баллов по результатам проведения аттестационного испытания в период промежуточной аттестации – 20 баллов приведенной шкалы.

Промежуточная аттестация по дисциплине проводится в соответствии с расписанием.

Итоговая оценка выводится на основании экзамена, перечень контрольных вопросов-тем приведен в разделе 11.2. Для этого составляются билеты по одному вопросу-теме в каждом. Экзамен проводится в письменной форме с последующим собеседованием-обсуждением с преподавателем, а также с рассмотрением результата самостоятельной работы студента в ходе практических занятий в виде сохраненных исходных текстов программ, работу которых студент демонстрировал ранее на занятиях. Также возможно повторное рассмотрение функционирование программ в ходе экзамена с применением компьютерных средств.

По предложенной теме студент должен продемонстрировать умение объяснить и обосновать сущность и свойства рассматриваемого механизма или концепции, указать в виде псевдокода или блок-схемы функционирование механизма. Также по допонительному вопросу преподавателя, по возможности, опрающегося на предоставленный студентом материал самостоятельной работы (листинг или интерактивно при помощи компьютера) найти в коде фрагменты, связанные с реализацией или использованием рассматрившаемго механизма или концепции.

В ходе собеседования возможно испльзование имеющейся компьютернрой техники для рассмтрения протокола работы программ или данных отладки в качестве иллюстрации к ответу или средтву сформулировать дополнительный вопрос.

Отвечая на вопрос и предоставляя материалы, студент должен продемонстрировать владение необходимыми программными средствами и умение работать с технической документацией по программированию, а также техническому описанию аппаратуры.

В случае, если рассматриваемый вопрос не находит отражения в предоставленных материалах, студенту предлагается в ходе экзамена построить либо блок-схему (псевдокод) для иллюстрации ответа, либо, используя имеющуюся компьютерную технику, самостоятельно построить набольшой пример, возможно, не обладающий полной функциональностью, но достаточный для оценки знаний студента.

Результаты промежуточной аттестации, определяются на основе баллов, набранных в рамках применения, СИД

Баллы по приведенной шкале в рамках применения СИД (ПСРО+ ПА)	**Оценка по результатам промежуточной аттестации**
	Экзамен/диф.зачет/зачет
0 - 60 баллов	Неудовлетворительно/не зачтено
61 - 75 баллов	Удовлетворительно/зачтено
76 - 89 баллов	Хорошо/зачтено
90 и более	Отлично/зачтено

11.2. Оценочные средства

Оценочные средства по дисциплине представлены в фонде оценочных средств, который является неотъемлемой частью основной образовательной программы и размещается в электронной информационно-образовательной среде СПбПУ на портале etk.spbstu.ru

12. Методические рекомендации по организации изучения дисциплины

Изучение дисциплины основываеся на поочередном изложении основных положений раздела и демонстрации рассмотренного аспекта на необльшом примере при помощи заранее заготовленных примеров программ и конфигурации окружения для их запуска, а также пошаговой отладки.

Для стабильного получения повторяемых результатов предполагается использование технологии виртуализации. И в ходе лекционных занятий и на упражнениях систематичсеки используется эмулятор (виртуальная машина) QEMU. Данный эмулятор имеет простые средства создания определенных конфигураций виртуальной аппаратуры и программный интерфейс к отладчику. Интерфейс отладчика позволяет загружать внешне подготовленные фрагменты кода, а также получать информацию о состоянии процессоров вычислительной системы. Сам эмулятор имеет монитор управления, при помощи которого можно получать информацию о состоянии виртуальных устройств, назначать их потоки данных на средства ввода-вывода и отображения информации хост компьютера, а также динамически добавлять и удалать устройства.

Также данная технология позволяет производить любые манипуляции с виртальными устройствами без угрозы повреждения важной информации и конфигурации хост системы.

Курс построен по принципу от нижнего уровня аппаратуры к верхним уровням программных абстракций. Соответсвенно, начальные занятия посвящены в основном изучению архитектуры аппаротных средств обеспечения ввода-вывода, отсновных концепций и механизмов обмена. Некоторые из рассматриваемых концепций уже знакомы студентам по дисциплине "Архитектура ЭВМ". Поэтому вводные занятия позволяют больше времени уделить самой применяемой технологии виртуализации и облегчить восприятие последующих тем с ее применением.

При представлении материала целесообразно использовать в качестве графического материала блок-схемы создаваемызх виртуальных конфигураций с указанием, какие связи какими командами управления конфигурацией эмулятора получнены, так как при работе в виртуальной среде студенты непосредтвенно виртуальную систему не видят. Для обозрения доступны только отображаемые символы потоков данных символоьных устройтсв и состояние регистров и памяти процессоров.

На занятиях по "аппаратным" темам используются небольшие фрагменты кода, демонстрирующие рассматриваемое поведение, например, передача символа, получение уроавления по сигралу прерывания, получение сетевого пакета в область памяти. При этом имеется возможность переиспозования кода, рассмотренного на предыдущем занятии как базы для следующего. Также объем кода от занятия к занятию может несколько возрастать. На

определенном этапе проводится реструктуризация кода в выносом уже рассмотренного кода в отдельные единицы трансляции и компоновка программы из нескольких единиц.

Переходным занятием от занятий, посвященным аппаратуре к занятиям, посвященным драйверам и модулям ядра является занятие, на котром рассмаривается механизм системных вызовов. Применене технологии виртуализации позволяет рассмотреть этот механизм в режиме пошаговой отладки. Также заготовленный приментр позволяент увидеть, как меняется режим работы процессора, и как работают механизмы защиты в режиме пользователя и ядра.

К момоенту изучения технологий разработки драйверов у студентов уже должно сформироваться представление о механизвах работы аппаратных средств и устройств вврода-вывода. Это позволит сосредоточиться на изучении архитектуры программного обеспечения.

На практических занятиях сначала следует дать студентам ознакомится с принеяемыми программными средствами создания кода программных модулей, созданием виртуальных конфигураций аппаратуры и объединение всех рассмотренных программных средств в единый процесс подготовки, моделирования и отладки.

По всем рассматриваемым темам в качестве задания для самостоятельной работы следует предлагагать расширить функциональность уже рассмотренных примеров, напрмер, добавить другое нарпоавление обмена данными, внести преобразование данных в определенном месте, например, при получении сигнала прерывания и т.п.

Также следует предлагать студентам самостоятельно освоить другие возможности программирования драйверов, например, освоить иное по сравнению с рассмотренным на занятиях устройстиво, присенить другие средства разработки программ, освоить другую программную среду или операционную систему, при услови, что она реализует схожие с рассматриваемыми в курсе концепции.

13. Адаптация рабочей программы для лиц с ОВЗ

Адаптированная программа разрабатывается при наличии заявления со стороны обучающегося (родителей, законных представителей) и медицинских показаний (рекомендациями психолого-медико-педагогической комиссии). Для инвалидов адаптированная образовательная программа разрабатывается в соответствии с индивидуальной программой реабилитации.