

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
«Гомельский государственный технический университет имени П.О.  
Сухого»

КАФЕДРА «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

РЕФЕРАТ

на тему

**ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ИМИТАЦИОННОГО  
МОДЕЛИРОВАНИЯ РОЖДЕНИЯ  $Z'$  - БОЗОНОВ В  
ПРОТОН-ПРОТОННЫХ СТОЛКНОВЕНИЯХ С УЧЕТОМ  
ЭФФЕКТОВ  $Z - Z'$  СМЕШИВАНИЯ**

подготовленный для прохождения итоговой аттестации по  
общеобразовательной дисциплине «Основы информационных технологи»

Выполнил:

магистрант гр. МАГ 40-12 специальности 1–40 80 04 «Математическое  
моделирование, численные методы и комплексы программ»

Бурим Илья Павлович

Проверил:

доцент кафедры «Информационные технологии»

Цитринов А.В.

Гомель 2017

# СОДЕРЖАНИЕ

<b>1</b>	<b>Операционная система Linux: состояние и тенденции развития</b>	<b>3</b>
1.1	Основные операционные системы . . . . .	3
1.2	Обзор Linux . . . . .	9
1.3	Сравнительный анализ Linux, MacOS и Windows . . . . .	13
1.4	Семейство Linux систем . . . . .	18
1.5	Нынешнее состояние и тенденции развития . . . . .	21
<b>2</b>	<b>Рождения <math>Z'</math> - бозонов в протон-протонных столкновениях с учетом эффектов <math>Z</math> - <math>Z'</math> смешивания</b>	<b>25</b>
2.1	Новая физика . . . . .	25
2.2	Инструменты имитационного моделирования . . . . .	30
2.3	Распады промежуточных бозонов. Определение массы $Z'$ -бозона . . . . .	32
	<b>ЗАКЛЮЧЕНИЕ</b>	<b>37</b>
	<b>Список использованных источников</b>	<b>38</b>

# ГЛАВА 1

## Операционная система Linux: состояние и тенденции развития

### 1.1 Основные операционные системы

В наше время существует огромное множество типов операционных систем, имеющих различные области применения. В таких условиях можно выделить четыре основных критерия, описывающих назначение ОС.

Операционная система (ОС) — комплекс взаимосвязанных программ, предназначенных для управления ресурсами вычислительного устройства. Благодаря этим программам происходит организация взаимодействия спользователем. Управление памятью, процессами, и всем программным и аппаратным обеспечением устраняет необходимость работы непосредственно с дисками и предоставляет простой, ориентированный на работу с файлами и интерфейс, скрывает множество неприятной работы с прерываниями, счетчиками времени, организацией памяти и другими компонентами. [1]

Организация удобного интерфейса, позволяющая пользователю взаимодействовать с аппаратурой компьютера за счет некой расширенной виртуальной машины, с которой удобнее работать и которую легче программировать. Вот перечень основных сервисов, предоставляемых типичными операционными системами. Разработка программ, где ОС представляет программисту разнообразные инструменты разработки приложений: редакторы, отладчики и т.п. Ему необязательно знать, как функционируют различные электронные и электромеханические узлы и устройства компьютера. Часто пользователь может обойтись только мощными высокоуровневыми функциями, которые представляет ОС. Также, для запуска программы нужно выполнить ряд действий: загрузить в основную память программу и данные, инициализировать устройства ввода-вывода и файлы, подготовить другие ресурсы. ОС выполняет всю эту работу вместо пользователя. ОС дает доступ к устройствам ввода-вывода. Каждое устройство требует свой набор команд для запуска. ОС предоставляет пользователю единообразный интерфейс, который опускает все детали и дает программисту доступ к устройствам ввода-вывода через простейшие команды чтения и записи. При работе с файлами управление со стороны ОС предполагает не только глубокий учет природы устройства ввода-вывода, но и знание структур данных, записанных в файлах. Многопользовательские ОС, кроме того,

обеспечивают механизм защиты при обращении к файлам. ОС управляет доступом к совместно используемой или общедоступной вычислительной системе в целом, а также к отдельным системным ресурсам. Она обеспечивает защиту ресурсов и данных от несанкционированного использования и разрешает конфликтные ситуации.

Обнаружение ошибок и их обработка – это еще один очень важный момент в назначении ОС. При работе компьютерной системы могут происходить разнообразные сбои за счет внутренних и внешних ошибок аппаратного обеспечения, различного рода программных ошибок (переполнение, попытка обращения к ячейке памяти, доступ к которой запрещен и др.). В каждом случае ОС выполняет действия, минимизирующие влияние ошибки на работу приложения (от простого сообщения об ошибке до аварийной остановки программы). И, наконец, учет использования ресурсов. ОС имеет средства учета использования различных ресурсов и отображения параметров производительности вычислительной системы. Эта информация важна для настройки (оптимизации) вычислительной системы с целью повышения ее производительности [1].

Организация эффективного использования ресурсов компьютера. ОС также является своеобразным диспетчером ресурсов компьютера. К числу основных ресурсов современных вычислительных систем относятся основная память, процессоры, таймеры, наборы данных, диски, накопители на магнитной ленте, принтеры, сетевые устройства, и др. Перечисленные ресурсы определяются операционной системой между выполняемыми программами. В отличие от программы, которая является статическим объектом, выполняемая программа – это динамический объект, который называется процессом и является базовым понятием современных ОС. Управление ресурсами вычислительной системы с целью наиболее эффективного их использования является вторым назначением операционной системы. Критерии эффективности, в соответствии с которыми ОС организует управление ресурсами компьютера, могут быть различными. Например, в одном случае наиболее важным является пропускная способность вычислительной системы, в другом – время ее реакции. Зачастую ОС должны удовлетворять нескольким, противоречащим друг другу критериям, что доставляет разработчикам серьезные трудности. Управление ресурсами включает решение ряда общих, не зависящих от типа ресурса задач. Планирование ресурса – определение процесса, для которого необходимо выделить ресурс. Здесь предопределяется, когда и в каком качестве должен выделяться данный ресурс. Удовлетворение запросов на ресурсы – выделение ресурсов процессам, мониторинг состояния и учет использования ресурса – поддержание оперативной информации о задействовании ресурса и использовании

его доли. Разрешение конфликтов между процессами, претендующими на один и тот же ресурс. Для решения этих общих задач управления ресурсами разные ОС используют различные алгоритмы, что в итоге и определяет облик ОС в целом, включая характеристики производительности, область применения и даже пользовательский интерфейс [1].

Облегчение процессов эксплуатации аппаратных и программных средств вычислительной системы. Ряд операционных систем имеет в своем составе наборы служебных программ, обеспечивающие резервное копирование, архивацию данных, проверку, очистку и дефрагментацию дисковых устройств и др. Кроме того, современные ОС имеют достаточно большой набор средств испособов диагностики и восстановления работоспособности системы. Сюда относятся: - диагностические программы для выявления ошибок в конфигурации операционной системы; - средства восстановления последней работоспособной конфигурации; - средства восстановления поврежденных и пропавших системных файлов и др. [1].

Современные ОС организуются таким образом, что допускают эффективную разработку, тестирование и внедрение новых системных функций, не прерывая процесса нормального функционирования вычислительной системы. Большинство операционных систем постоянно развиваются (нагляден пример *Windows*). Происходит это в силу следующих причин [1]. Для удовлетворения пользователей или нужд системных администраторов ОС должны постоянно предоставлять новые возможности. Например, может потребоваться добавить новые инструменты для контроля или оценки производительности, новые средства ввода-вывода данных (речевой ввод). Другой пример – поддержка новых приложений, использующих окна на экране дисплея [1]. В каждой ОС есть ошибки. Время от времени они обнаруживаются и исправляются. Отсюда постоянные появления новых версий и редакций ОС. Необходимость регулярных изменений накладывает определенные требования на организацию операционных систем. Очевидно, что эти системы должны иметь модульную структуру с четко определенными межмодульными связями. Важную роль играет хорошая и полная документированность системы [1].

Функции ОС обычно группируются либо в соответствии с типами локальных ресурсов, которыми управляет ОС, либо в соответствии со специфическими задачами, применимыми ко всем ресурсам. Совокупности модулей, выполняющих такие группы функций, образуют подсистемы операционной системы. Наиболее важными подсистемами управления ресурсами являются подсистемы управления процессами, памятью, файлами и внешними устройствами, а подсистемами, общими для всех ресурсов, являются подсистемы пользовательского интерфейса, защиты данных и администри-

рования [2].

Подсистема управления процессами непосредственно влияет на функционирование вычислительной системы. Для каждой выполняемой программы ОС организует один или более процессов. Каждый такой процесс представляется в ОС информационной структурой (таблицей, дескриптором, контекстом процессора), содержащей данные о потребностях процесса в ресурсах, а также о фактически выделенных ему ресурсах (область оперативной памяти, количество процессорного времени, файлы, устройства ввода-вывода и др.). В современных мультипрограммных ОС может существовать одновременно несколько процессов, порожденных по инициативе пользователей и их приложений, а также инициированных ОС для выполнения своих функций (системные процессы). Поскольку процессы могут одновременно претендовать на одни и те же ресурсы, подсистема управления процессами планирует очередность выполнения процессов, обеспечивает их необходимыми ресурсами, обеспечивает взаимодействие и синхронизацию процессов [2].

Подсистема управления памятью производит распределение физической памяти между всеми существующими в системе процессами, загрузку и удаление программных кодов и данных процессов в отведенные им области памяти, а также защиту областей памяти каждого процесса. Стратегия управления памятью складывается из стратегий выборки, размещения и замещения блока программы или данных в основной памяти. Соответственно используются различные алгоритмы, определяющие, когда загрузить очередной блок в память, в какое место памяти его поместить и какой блок программы или данных удалить из основной памяти, чтобы освободить место для размещения новых блоков. Одним из наиболее популярных способов управления памятью в современных ОС является виртуальная память. Реализация механизма виртуальной памяти позволяет программисту считать, что в его распоряжении имеется однородная оперативная память, объем которой ограничивается только возможностями адресации, предоставляемыми системой программирования.

Нарушения защиты памяти связаны с обращениями процессов к участкам памяти, выделенной другим процессам прикладных программ или программ самой ОС. Средства защиты памяти должны пресекать такие попытки доступа путем аварийного завершения программы-нарушителя.

Функции управления файлами сосредоточены в файловой системе ОС. Операционная система виртуализирует отдельный набор данных, хранящихся на внешнем накопителе, в виде файла – простой неструктурированной последовательности байтов, имеющих символьное имя. Для удобства работы с данными файлы группируются в каталоги, которые, в свою оче-

редь, образуют группы – каталоги более высокого уровня. Файловая система преобразует символьные имена файлов, с которыми работает пользователь или программист, в физические адреса данных на дисках, организует совместный доступ к файлам, защищает их от несанкционированного доступа [2].

Функции управления внешними устройствами возлагаются на подсистему управления внешними устройствами, называемую также подсистемой ввода-вывода. Она является интерфейсом между ядром компьютера и всеми подключенными к нему устройствами. Спектр этих устройств очень обширен (принтеры, сканеры, мониторы, модемы, манипуляторы, сетевые адаптеры, АЦП разного рода и др.), сотни моделей этих устройств отличаются набором и последовательностью команд, используемых для обмена информацией с процессором и другими деталями. Программа, управляющая конкретной моделью внешнего устройства и учитывающая все его особенности, называется драйвером. Наличие большого количества подходящих драйверов во многом определяет успех ОС на рынке. Созданием драйверов занимаются как разработчики ОС, так и компании, выпускающие внешние устройства. ОС должна поддерживать четко определенный интерфейс между драйверами и остальными частями ОС. Тогда разработчики компаний-производителей устройств ввода-вывода могут поставлять вместе со своими устройствами драйверы для конкретной операционной системы [2].

Безопасность данных вычислительной системы обеспечивается средствами отказоустойчивости ОС, направленными на защиту от сбоев и отказов аппаратуры и ошибок программного обеспечения, а также средствами защиты от несанкционированного доступа. Для каждого пользователя системы обязательна процедура логического входа, в процессе которой ОС убеждается, что в систему входит пользователь, разрешенный административной службой. Корпорация *Microsoft*, например, в своем последнем продукте *Windows 10* предлагает пользователю вход в систему через распознавание внешности. Это должно повысить безопасность и сделать вход в систему быстрее [3]. А вот *Google* обещает нам в новой версии своей ОС для смартфонов *Android 6.0* доступ к устройству и подтверждение покупок через сканер отпечатка пальца, если для того пригодно устройство. Администратор вычислительной системы определяет и ограничивает возможности пользователей в выполнении тех или иных действий, т.е. определяет их права по обращению и использованию ресурсов системы. Важным средством защиты являются функции аудита ОС, заключающиеся в фиксации всех событий, от которых зависит безопасность системы. Поддержка отказоустойчивости вычислительной системы реализуется на

основе резервирования (дисковые *RAID*-массивы, резервные принтеры и другие устройства, иногда резервирование центральных процессоров, в ранних ОС – дуальные и дуплексные системы, системы с мажоритарным органом и др.). Вообще обеспечение отказоустойчивости системы – одна из важнейших обязанностей системного администратора, который для этого использует ряд специальных средств и инструментов [2].

Прикладные программисты используют в своих приложениях обращение к операционной системе, когда для выполнения тех или иных действий им требуется особый статус, которым обладает только ОС. Возможности операционной системы доступны программисту в виде набора функций, который называется интерфейсом прикладного программирования (*Application Programming Interface, API*). Приложения обращаются к функциям *API* с помощью системных вызовов. Способ, которым приложение получает услуги операционной системы, очень похож на вызов подпрограмм. Способ реализации системных вызовов зависит от структурной организации ОС, особенностей аппаратной платформы и языка программирования. В ОС *UNIX* системные вызовы почти идентичны библиотечным процедурам [1].

ОС обеспечивает удобный интерфейс не только для прикладных программ, но и для пользователя (программиста, администратора, пользователя). На данный момент производители предлагают множество функций, призванных облегчить работу с устройствами и сэкономить время. В качестве примера *Windows 10. Microsoft* помогает пользователю обеспечить беспрепятственную работу всех его устройств (естественно от *Microsoft*), за счет общей ОС. Тут и мгновенная передача данных с одного устройства на другое, и общие уведомления, которые с такой функцией никак не пропустишь [3]. «Эффективная, организованная работа» – это практически слоган для каждого производителя ОС. Работа с заметками прямо на веб-страницах, новые многооконные режимы, несколько рабочих столов – все это уже есть несколько лет, а у разработчиков еще много идей [3].

Современные операционные системы имеют сложную структуру, состоящую из множества элементов, где каждый из них выполняет определенные функции по управлению процессами и распределению ресурсов. Ядро ОС – центральная часть операционной системы, обеспечивающая приложениям координированный доступ к файловой системе, и обмен файлами между ПУ [2].

Командный процессор. Программный модуль ОС, ответственный за чтение отдельных команд или же последовательности команд из командного файла, иногда называют командным интерпретатором.

Драйверы устройств. К магистрали компьютера подключаются различ-



ные устройства (дисководы, монитор, клавиатура, мышь, принтер и др.). Каждое устройство выполняет определенную функцию, при этом техническая реализация устройств существенно различается. В состав операционной системы входят драйверы устройств, специальные программы, которые обеспечивают управление работой устройств и согласование информационного обмена с другими устройствами, а также позволяют производить настройку некоторых параметров устройств. Каждому устройству соответствует свой драйвер.

Утилиты. Дополнительные сервисные программы (утилиты) – вспомогательные компьютерные программы в составе общего программного обеспечения, делающие удобным и многосторонним процесс общения пользователя с компьютером.

Для удобства пользователя в состав операционной системы обычно входит также справочная система. Справочная система позволяет оперативно получить необходимую информацию как о функционировании операционной системы в целом, так и о работе ее отдельных модулей [2].

## 1.2 Обзор Linux

*Linux* — общее название *UNIX*-подобных операционных систем на основе одноимённого ядра и собранных для него библиотек и системных программ, разработанных в рамках проекта *GNU*. Система *UNIX* приобрела популярность в связи с ее успешным использованием на мини-ЭВМ. Этот успех послужил толчком к тому, чтобы создать подобную систему и для персональных компьютеров. Как правило, различные версии ОС, относящиеся к этому семейству, имеют свои названия, но в основных чертах повторяют особенности *UNIX*.

*GNU/Linux* работает на *PC*-совместимых системах семейства *Intel x86*, а также на *IA-64*, *AMD64*, *PowerPC*, *ARM* и многих других.

*UNIX* - операционная система, которая позволяет осуществить выполнение работ в многопользовательском и многозадачном режиме. Поначалу она предназначалась для больших ЭВМ, чтобы заменить *MULTICS*. *UNIX* является очень мощным средством в руках программиста, но требует очень большого объёма ОЗУ и пространства диска. Несмотря на попытки стандартизировать эту операционную систему, существует большое количество различных его версий, главным образом потому, что она была распространена в виде программы на языке Си, которую пользователи стали модифицировать для своих собственных нужд.

Очень многие считают, что *Linux* - это только ядро. Но одно только ядро бесполезно для пользователя. Хотя ядро, несомненно, основа ОС *Linux*,

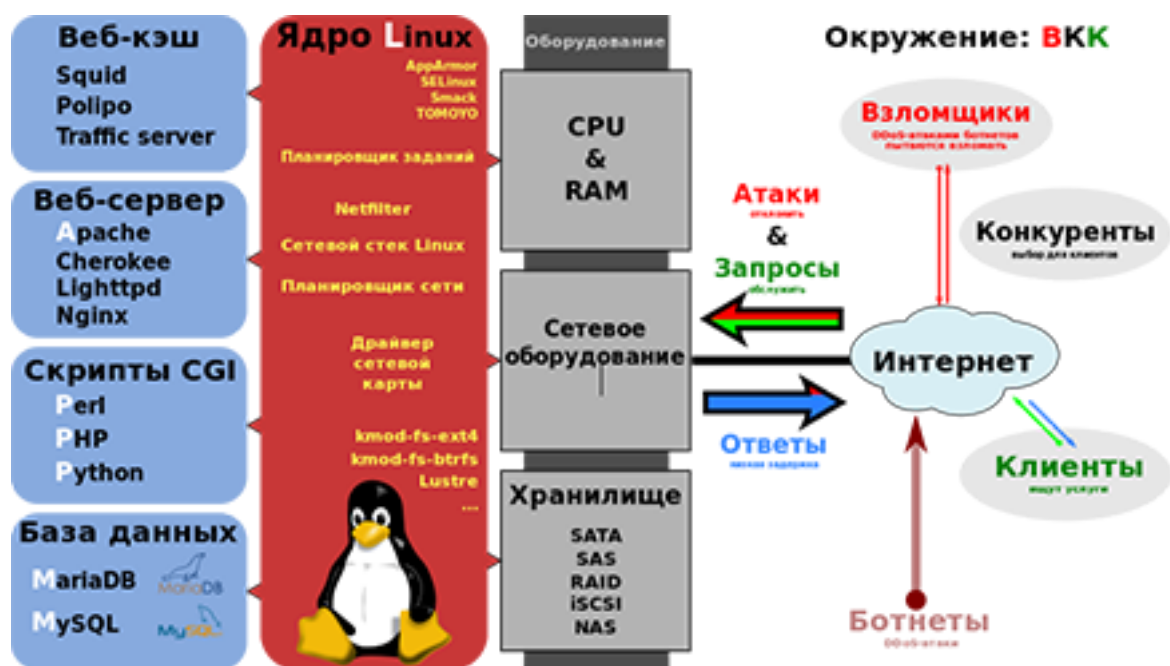


Рисунок 1.1 — Структура *Linux*-системы

пользователю все время приходится работать с прикладными программами. Эти программы не менее важны, чем ядро. Поэтому *Linux* - это совокупность ядра и основных прикладных программ, которые обычно бывают установлены на каждом компьютере с этой операционной системой. Объединение ядра и прикладных программ в единое целое проявляется и в названии системы: *GNU/Linux*. *GNU* - это проект по созданию комплекса программ, подобного тому, что обычно сопровождает *Unix*-подобную систему.

*Linux* изначально был написан Линусом Торвальдсом, а затем улучшался бесчисленным количеством народа во всем мире. Он является клоном операционной системы *Unix*, одной из первых мощных операционных систем, разрабатываемых для компьютеров, но не бесплатной. Но ни *Unix System Laboratories* (создатели *Unix*), ни Университет Беркли, разработчики *Berkeley Software Distribution (BSD)*, не участвовали в его создании. Один из наиболее интересных фактов из истории *Linux*'а - это то, что в его создании принимали участие одновременно люди со всех концов света - от Австралии до Финляндии - и продолжают это делать до сих пор.

Дистрибутив - это сама ОС плюс набор пакетов программ для *Linux*. Стоит также упомянуть, что все это поставляется с исходными текстами, и любую программу, написанную под *Linux*, можно переделать под себя. Это же позволяет перенести любую программу на любую платформу - *Intel PC*, *Macintosh*. Все вышеописанное получилось благодаря *Free Software*

*Foundation*, фонду бесплатных программ, который является частью проекта *GNU*. И именно для этих целей была создана *General Public License (GPL)*, исходя из которой, *Linux* – бесплатен, как и весь софт под него, причем коммерческое использование программного обеспечения для *Linux* или его частей запрещено.

Кроме всего ранее упомянутого, *Linux* – очень мощная и стабильная ОС. Использование его в Сети оправдывает себя, да и взломать его не так уж и легко.

На сегодняшний день, развитие *Linux* идет по двум ветвям. Первая, с четными номерами версий (2.0, 2.2, 2.4), считается более стабильной, надежной версией *Linux*. Вторая, чьи версии нумеруются нечетными номерами (2.1, 2.3), является более дерзкой и быстрее развивающейся и, следовательно (к сожалению), более богатой ошибками.

В *Linux* нет разделения на диски *C*, *D*, и процесс общения с устройствами очень удобен. Все устройства имеют собственный системный файл, все диски подключаются к одной файловой системе и выглядит это все как бы монолитно, едино. Четкая структура каталогов позволяет находить любую информацию мгновенно. Для файлов библиотек – свой каталог, для запускаемых файлов – свой, для файлов с настройками – свой, для файлов устройств – свой, и так далее.

Модульность ядра позволяет подключать любые сервисы ОС без перезагрузки компьютера. Кроме того, вы можете переделать само ядро ОС, благо исходные тексты ядра также имеются в любом дистрибутиве.

В ОС *Linux* очень умело, если так можно выразиться, используется идея многозадачности, т.е. любые процессы в системе выполняются одновременно (сравните с *Windows*: копирование файлов на дискету и попытка слушать в этот момент музыку не всегда совместимы).

*Linux* чуть более сложен, чем *Windows*, и не всем так просто перейти на него после использования окон. На первый взгляд, может даже показаться, что он очень неудобен и труднонастраиваем.

*Linux* можно настроить под себя, настроить так, что от пользования этой ОС вы будете испытывать огромное удовлетворение. Огромное количество настроек позволяет изменить внешний (да и внутренний) вид ОС. В *Linux* есть выбор в использовании графической оболочки, есть несколько офисных пакетов, программы-серверы, файрволы.

В 1998 *Linux* была самой быстро развивающейся операционной системой для серверов, распространение которой увеличилось в том же году на 212 %.

Сегодня пользователей *Linux* насчитывается более 20,000,000. Под *Linux* существует множество приложений, предназначенных как для домашнего

использования, так и для полностью функциональных рабочих станций *UNIX* и серверов *Internet*.

Если при использовании коммерческой операционной системы пользователь вынужден ждать выхода следующей версии для того, чтобы получить систему без недостатков предыдущей версии, то модульность Линукса позволяет скачать новое ядро, которое выходит не реже раза в два месяца, а то и чаще (стабильная версия). В Линукс-системах пользователи работают через интерфейс командной строки (*CLI*), графический интерфейс пользователя (*GUI*), или, в случае встраиваемых систем, через элементы управления соответствующих аппаратных средств.

Настольные системы, как правило, имеют графический пользовательский интерфейс, в котором командная строка доступна через окно эмулятора терминала или в отдельной виртуальной консоли. Большинство низкоуровневых компонентов Линукс, включая пользовательские компоненты *GNU*, используют исключительно командную строку.

Командная строка особенно хорошо подходит для автоматизации повторяющихся или отложенных задач, а также предоставляет очень простой механизм межпроцессного взаимодействия. Программа графического эмулятора терминала часто используется для доступа к командной строке с рабочего стола Линукс.

Линукс-системы обычно реализуют интерфейс командной строки при помощи оболочки операционной системы, которая также является традиционным способом взаимодействия с системой *Unix*. Дистрибутивы, специально разработанные для серверов, могут использовать командную строку в качестве единственного интерфейса.

На настольных системах наибольшей популярностью пользуются пользовательские интерфейсы, основанные на таких средах рабочего стола как *KDE Plasma Desktop*, *GNOME* и *Xfce*, хотя также существует целый ряд других пользовательских интерфейсов. Самые популярные пользовательские интерфейсы основаны на *X Window System* (часто просто «*X*» или «иксы»). «*X*» предоставляет прозрачность сети и позволяет графическим приложениям, работающим на одном компьютере, отображаться на другом компьютере, на котором пользователь может взаимодействовать с ними.

Другие графические интерфейсы, такие как *FVWM*, *Enlightenment* и *Window Maker*, могут быть классифицированы как простые менеджеры окон *X Window System*, которые предоставляют окружение рабочего стола с минимальной функциональностью.

Оконный менеджер предоставляет средства для управления размещением и внешним видом отдельных окон приложений, а также взаимодействует с *X Window System*. Окружение рабочего стола включает в себя окон-

ные менеджеры, как часть стандартной установки: (*Metacity* для *GNOME*, *KWin* для *KDE*, *Xfwm* для *Xfce* с 2010 года), хотя пользователь при желании может выбрать другой менеджер окон.

*Linux* работает на множестве процессоров различных архитектур, таких как *x86*, *x86-64*, *PowerPC*, *ARM*, *Alpha AXP*, *SPARC*, *Motorola 680x0*, *SuperH*, *IBM System/390*, *MIPS*, *PA-RISC*, *AXIS CRIS*, *Renesas M32R*, *Atmel AVR32*, *Renesas H8/300*, *NEC V850*, *Tensilica Xtensa* и многих других.

В отличие от коммерческих систем, таких как *Windows* или *Mac OS X*, *Linux* не имеет географического центра разработки. Нет и организации, которая владела бы этой системой. Нет даже единого координационного центра. Программы для *Linux* — результат работы тысяч проектов. Некоторые из этих проектов централизованы, некоторые сосредоточены в фирмах. Многие проекты объединяют хакеров со всего света, которые знакомы только по переписке. Создать свой проект или присоединиться к уже существующему может любой и, в случае успеха, результаты работы станут известны миллионам пользователей. Пользователи принимают участие в тестировании свободных программ, общаются с разработчиками напрямую, что позволяет быстро находить и исправлять ошибки и реализовывать новые возможности.

### 1.3 Сравнительный анализ Linux, MacOS и Windows

Главной отличительной чертой *Linux* системы является ее модульность и обширный набор системных программ, которые позволяли создать благоприятную обстановку для пользователей-программистов. Система *UNIX* органически сочетается с языком Си, на котором написано более 90% ее собственных модулей. Командный язык системы практически совпадает с языком Си, что позволяло очень легко комбинировать различные программы при создании больших прикладных систем.

*UNIX* имеет «оболочку», с которой пользователь непосредственно взаимодействует, и «ядро», которое, собственно, и управляет действиями компьютера. Компьютер выводит в качестве приглашения для ввода команд долларový знак. Из-за продолжительности пользования этой операционной системы количество команд весьма велико. В дополнение к командам по управлению файлами, которые присутствуют в любой операционной системе, *UNIX* имеет, по крайней мере, один текстовый редактор, а также форматор текста и компилятор языка Си, что позволяет, по мере надобности, модифицировать «оболочку».

От *UNIX* многие другие операционные системы переняли такие функ-

ции, как переназначение, канал и фильтр; однако *UNIX* имеет, несомненно, преимущество в том, что она с самого начала разрабатывалась как многопользовательская и многозадачная операционная система. Имена файлов могут иметь 14 знаков, причём в именах файлов различаются заглавные и строчные буквы. Первоначальный набор команд операционной системы расширился до 143 в версии 7.0; в версии *System III* добавилась ещё 71 команда, ещё 25 - в *Berkeley 4.1* и следующие 114 в *Berkeley 4.2*. Из-за такого обилия команд *UNIX* не относится к самым удобным для пользователя языкам. Работа облегчается, если применять графический пользовательский интерфейс, но поскольку такое количество команд и без того занимает значительный объём памяти, этот интерфейс требует ещё большего объёма памяти и пространства диска.

С тем, что такое операционные системы и их особенностями в целом, мы разобрались, теперь самое время приступить к более детальному, конкретному рассмотрению многообразия ОС, которое обычно начинается с рассмотрения краткой истории появления и развития

Считается, что в появлении Юникса в частности виновата компьютерная игра. Дело в том, что Кен Томпсон непонятно чего ради создал игрушку «*Space Travel*». Он написал ее в 1969 году на компьютере *Honeywell-635*, который использовался для разработки *Multics*. Но фишка в том, что ни вышеупомянутый *Honeywell*, ни имевшийся в лаборатории *General Electric-645* не подходили для игрушки. И Кену пришлось найти другую ЭВМку - 18-разрядный компьютер *PDP-7*. Кен с ребятами разрабатывал новую файловую систему, дабы облегчить себе жизнь и работу. Ну и решил опробовать свое изобретение на новенькой машине. Опробовал. Весь отдел патентов *Bell Labs* дружно радовался. Томпсону этого показалось мало и он начал ее усовершенствовать, включив такие функции как *inodes*, подсистему управления процессами и памятью, обеспечивающую использование системы двумя пользователями в режиме *TimeSharing* (разделения времени) и простой командный интерпретатор. Кен даже разработал несколько утилит под систему. Собственно, сотрудники Кена еще помнили, как они мучались над ОС *Multics*, поэтому в честь старых заслуг один из них - Брайан Керниган - решил назвать ее похожим именем - *UNICS*. Через некоторое время название сократили до *UNIX* (читается так же, просто писать лишнюю букву настоящим программистам во все времена было лень). ОС была написана на ассемблере. Вот мы и подбираемся к тому, что известно в мире как «Первая редакция *UNIX*». В ноябре 1971 года был опубликован первый выпуск полноценной доки по Юниксу. В соответствии с этим и ОС была названа «Первой редакцией *UNIX*». Вторая редакция вышла довольно быстро - меньше, чем через год. Третья редакция ничем особенным

не отличалась. Разве что заставила Дениса Ритчи «засесть за словари», вследствие чего тот написал собственный язык, известный сейчас как *C*. Именно на нём была написана 4-я редакция *UNIX* в 1973 году. В июле 1974 года вышла 5-я версия *UNIX*. Шестая редакция *UNIX* (*UNIX V6*), выпущенная в 1975 году, стала первым коммерчески распространяемым Юниксом. Большая ее часть была написана на Си.

Позже была полностью переписана подсистема управления оперативной и виртуальной памятью, заодно изменили интерфейс драйверов внешних устройств. Все это позволило сделать систему легко переносимой на другие архитектуры и было названо «Седьмая редакция» (*UNIX version 7*). Когда в 1976 году в Университет Беркли попала «шестерка», там возникли местные Юникс-гуру. Одним из них был Билл Джой.

Собрав своих друзей-программистов, Билли начал разработку собственной системы на ядре *UNIX*. Запихнув помимо основных функций кучу своих (включая, компилятор Паскаля), он назвал всю эту сборную солянку *Distribution (BSD 1.0)*. Вторая версия *BSD* почти ни чем не отличалась от первой. Третья версия *BSD* основывалась на переносе *UNIX Version 7* на компьютеры семейства *VAX*, что дало систему *32/V*, легшую в основу *BSD 3.x*. Ну, и самое главное - при этом был разработан стек протоколов *TCP/IP*; разработка финансировалась Министерством Безопасности США.

Первая коммерческая система называлась *UNIX SYSTEM III* и вышла она в 1982 году. В этой ОС сочетались лучшие качества *UNIX Version 7*. Далее Юниксы развивались примерно так: во-первых, появились компании, занимавшиеся коммерческим переносом *UNIX* на другие платформы. К этому приложила руку и небезызвестная *Microsoft Corporation*, совместно с *Santa Cruz Operation* произведшая на свет *UNIX*-вариацию под названием *XENIX*, во-вторых, *Bell Labs* создала группу по развитию Юникса и объявила о том, что все последующие коммерческие версии *UNIX* (начиная с *System V*) будут совместимы с предыдущими.

К 1984-му году был выпущен второй релиз *UNIX System V*, в котором появились: возможности блокировок файлов и записей, копирования совместно используемых страниц оперативной памяти при попытке записи (*copy-on-write*), страничного замещения оперативной памяти и т. д. К этому времени ОС *UNIX* была установлена на более чем 100 тыс. компьютеров.

В 1987-м году выпущен третий релиз *UNIX System V*. Было зарегистрировано четыре с половиной миллиона пользователей этой эпической операционной системы. Что касается *Linux*, то он возник лишь в 1990 году, а первая официальная версия ОС вышла лишь в октябре 1991. Как и *BSD*,

*Linux* распространялся с исходниками, чтобы любой пользователь мог настроить ее себе так, как ему хочется. Настраивалось практически всё, чего не может себе позволить, например, *Windows 9x*.

Разработчиком *MacOS* является фирма *Apple* - законодатель мод в области *GUI*, начиная с 1980-х гг. Ключевой идеей *MacOS* с самого начала является разработка и развитие ОС только на основе графического пользовательского интерфейса - «ОС без командной строки». Аппаратная платформа *MacOS* – всевозможные семейства компьютеров *Macintosh* фирмы *Apple* (наиболее популярные среди рабочих станций в США), а также *PowerPC* – рабочая станция *RISC*-архитектуры, совместно разработанная *Apple*, *IBM* и *HP*. Диалекты (версии) *MacOS* различаются по своему подходу к реализации, хотя для пользователя, благодаря удобному графическому интерфейсу, эти различия могут быть незаметны. Классическая *MacOS* (*classic MacOS*) - оригинальная разработка фирмы *Apple*; новая линия *MacOS X* – развитие ОС *MacOS Classic* и ОС *NeXTSTEP* (*UNIX*-подобной ОС), т.е. она является *UNIX*-совместимой.

Для компьютеров *Macintosh* характерен способ соединения устройств *Apple Desktop bus* (сокращённо *ADB*) – специальный вид соединений для подключения мыши, клавиатуры и джойстика к компьютеру. При его использовании все устройства могут соединяться между собой, и только одно следует подключить к машине непосредственно. Такой способ подключения называется цепочкой. Также возможно последовательное соединение (*Daisy chaining*) ряда дополнительных устройств, таких как проигрыватель компакт-дисков, жёсткий диск и сканер.

*Macintosh*-машины имеют встроенную поддержку звука и возможность подключения к сети, а также обеспечивают функцию «включи и работай» (*plug and play*), т.е. при подключении нового оборудования необходимо только произвести его наладку и согласовать с имеющимся. Компьютеры *Macintosh Power* или *Macintosh AV* при помощи адаптера *Geo Port* соединяются непосредственно с телефонной линией, позволяя связываться с оперативными службами, получать и передавать факсы без использования модема.

Персональные компьютеры *Macintosh* представлены несколькими видами. Компактные (моноблочные) модели целиком находятся в одном корпусе. Они могут иметь встроенный или внешний винчестер, у них четкие, но малые по размеру экраны, не отображающие цвета или полутона, мало оперативной памяти и нет гнезд для плат расширения. Модульные *Macintosh*, в отличие от компактных, состоят из нескольких частей или модулей. Все процессорные блоки у таких моделей имеют одно или несколько гнезд расширения (слотов). Мобильные (портативные) *Macintosh* ти-



па *Power book* представляют собой миниатюрные, легко транспортируемые модели. Небольшие размеры машин этой серии, однако, ограничивают возможности и могут способствовать медленной работе. Вместо мышки (*mouse*) мобильные модели имеют встроенные шаровые манипуляторы – колобки (*trackball*) или чувствительную к перемещению пальца площадку – сенсорный планшет или пятачок (*trackpad*), что позволяет экономить место.

С момента выпуска первого компьютера *Macintosh* в 1984 году на всех машинах этого семейства и совмещенных с ними используется операционная система *Mac OS*, которая, конечно, может быть дополнена. Параметры её варьируются от версии к версии. Для функционирования компьютера наиболее необходимой является «Системная Папка» (*System Folder*), которая содержит всю основную информацию, задающую систему *Mac Os*. В ней находятся *System file*, *Finder*, шрифты, реквизиты (прикладные мини-программы, а также другие часто используемые программы), принтерные шрифты и т.д., она всегда выделяется особой пиктограммой. Из неё можно легко установить необходимые программы.

*Windows* была, наверное, первой операционной системой, которую Биллу Гейтсу никто не заказывал, а разрабатывать ее он взялся на свой страх и риск. Что в ней такого особенного? Во-первых, графический интерфейс. Такой на тот момент был только у пресловутой *Mac OS*. Во-вторых, многозадачность. В общем, в ноябре 1985 вышла *Windows 1.0*. Основной платформой ставились 286-е машины.

Ровно через два года, в ноябре 87-го вышла *Windows 2.0*, через полтора года вышла 2.10. Ничего особенного в них не было. И вот, наконец, революция! Май 1990-го года, вышла *Windows 3.0*. Чего там только не было: и ДОС-приложения выполнялись в отдельном окне на полном экране, и *Copy-Paste* работал для обмена с данными ДОС - приложений, и сами *Windows* работали в нескольких режимах памяти: в реальном (базовая 640 Кб), в защищенном и расширенном. При этом можно было запускать приложения, размер которых превышает размер физической памяти. Имел место быть и динамический обмен данными (*DDE*). Через пару лет вышла и версия 3.1, в которой уже отсутствовали проблемы с базовой памятью. Также была введена новомодная функция, поддерживающая шрифты *True Type*. Обеспечена нормальная работа в локальной сети. Появился *Drag&Drop* (перенос мышкой файлов и директорий). В версии 3.11 была улучшена поддержка сети и введено еще несколько малозначительных функций. Параллельно вышла *Windows NT 3.5*, которая была на тот момент сбором основных сетевых примочек, взятых из *OS/2*. В июне 1995 вся компьютерная общественность была взбодорена сообщением Microsoft о

релизе в августе новой операционной системы, существенно иной, нежели *Windows 3.11*.

24 августа – дата официального релиза *Windows 95* (другие названия: *Windows 4.0*, *Windows Chicago*). Теперь это была не просто операционная среда – это была полноценная операционная система. 32-битное ядро позволяло улучшить доступ к файлам и сетевым функциям. 32-битные приложения были лучше защищены от ошибок друг друга, имелась и поддержка многопользовательского режима на одном компьютере с одной системой. Множество отличий в интерфейсе, куча настроек и улучшений.

Чуть позже вышла новая *Windows NT* с тем же интерфейсом, что и 95-е. Поставлялась в двух вариантах: как сервер и как рабочая станция. Системы *Windows NT 4.x* были надежны, но не столько потому, что у *Microsoft* проснулась совесть, сколько потому, что *NT* писали программисты, когда-то работавшие над *VAX/VMS*.

## 1.4 Семейство Linux систем

К операционной системе *GNU/Linux* также часто относят программы, дополняющие эту операционную систему, и прикладные программы, делающие её полноценной многофункциональной операционной средой. В отличие от большинства других операционных систем, *GNU/Linux* не имеет единой «официальной» комплектации. Вместо этого *GNU/Linux* поставляется в большом количестве так называемых дистрибутивов, в которых программы *GNU* соединяются с ядром *Linux* и другими программами.

В отличие от *Microsoft Windows*, *Mac OS* и коммерческих UNIX-подобных систем, *GNU/Linux* не имеет географического центра разработки. Нет и организации, которая владела бы этой системой, нет даже единого координационного центра. Программы для *Linux* — результат работы тысяч проектов. Некоторые из этих проектов централизованы, некоторые сосредоточены в фирмах. Многие проекты объединяют хакеров со всего света, которые знакомы только по переписке. Создать свой проект или присоединиться к уже существующему может любой и, в случае успеха, результаты работы станут известны миллионам пользователей. Пользователи принимают участие в тестировании свободных программ, общаются с разработчиками напрямую, что позволяет быстро находить и исправлять ошибки и реализовывать новые возможности.

Именно такая гибкая и динамичная система разработки, невозможная для проектов с закрытым кодом, определяет исключительную экономическую эффективность *GNU/Linux*. Низкая стоимость свободных разработок, отлаженные механизмы тестирования и распространения, привлечение

людей из разных стран, обладающих разным видением проблем, защита кода лицензией *GPL* — всё это стало причиной успеха свободных программ.

Конечно, такая высокая эффективность разработки не могла не заинтересовать крупные фирмы, которые стали открывать свои проекты. Так появились *Mozilla* (*Netscape*, *AOL*), *OpenOffice.org* (*Sun*), свободный клон *Interbase* (*Borland*) — *Firebird*, *SAP DB* (*SAP*). *IBM* способствовала переносу *GNU/Linux* на свои мейнфреймы.

С другой стороны, открытый код значительно снижает себестоимость разработки закрытых систем для *GNU/Linux* и позволяет снизить цену решения для пользователя. Вот почему *GNU/Linux* стала платформой, часто рекомендуемой для таких продуктов, как *Oracle*, *DB2*, *Informix*, *SyBase*, *SAP R3*, *Domino*.

Большинство пользователей для установки *GNU/Linux* используют дистрибутивы. Дистрибутив — это не просто набор программ, а ряд решений для разных задач пользователей, объединённых едиными системами установки, управления и обновления пакетов, настройки и поддержки. [3]

Самые распространённые в мире дистрибутивы:

- *Ubuntu*. Быстро завоевавший популярность дистрибутив, ориентированный на лёгкость в освоении и использовании.
- *openSUSE*. Бесплатно распространяемая версия дистрибутива SuSE, принадлежащая компании *Novell*. Отличается удобством в настройке и обслуживании благодаря использованию утилиты *YaST*.
- *Fedora*. Поддерживается сообществом и корпорацией *RedHat*, предшествует выпускам коммерческой версии *RHEL*.
- *Debian*. Международный дистрибутив, разрабатываемый обширным сообществом разработчиков в некоммерческих целях. Послужил основой для создания множества других дистрибутивов. Отличается строгим подходом к включению несвободного ПО.
- *Mandriva*. Французско-бразильский дистрибутив, объединение бывших *Mandrake* и *Conectiva*. *Slackware*. Один из старейших дистрибутивов, отличается консервативным подходом в разработке и использовании.
- *Gentoo*. Дистрибутив, собираемый из исходных кодов. Позволяет очень гибко настраивать конечную систему и оптимизировать производительность, поэтому часто называет себя мета-дистрибутивом. Ориентирован на экспертов и опытных пользователей.

- *Archlinux*. Ориентированный на применение самых последних версий программ и постоянно обновляемый, поддерживающий одинаково как бинарную, так и установку из исходных кодов и построенный на философии простоты «*KISS*» («*Keep it simple, stupid*» / «Не усложняй»), этот дистрибутив ориентирован на компетентных пользователей, которые хотят иметь всю силу и модифицируемость *Linux*, но не в жертву времени обслуживания.

Помимо перечисленных, существует множество других дистрибутивов, как базирующихся на перечисленных, так и созданных с нуля и зачастую предназначенных для выполнения ограниченного количества задач.

Каждый из них имеет свою концепцию, свой набор пакетов, свои достоинства и недостатки. Ни один не может удовлетворить всех пользователей, а потому рядом с лидерами благополучно существуют другие фирмы и объединения программистов, предлагающие свои решения, свои дистрибутивы, свои услуги. Существует множество *LiveCD*, построенных на основе *GNU/Linux*, например, *Knoppix*. *LiveCD* позволяет запускать *GNU/Linux* непосредственно с компакт-диска, без установки на жёсткий диск. Большинство крупных дистрибутивов, включая *Ubuntu*, могут быть использованы как *LiveCD*.

Дистрибутивы *Linux* часто бывают ориентированы на конкретные задачи. Поэтому не получится просто составить список операционных систем и сказать: «они – самые лучшие». Здесь выделены несколько областей использования *Linux* и выбраны те дистрибутивы, у которых есть все шансы стать первыми в своей нише в 2017-м. [4]

Лучший дистрибутив для системных администраторов это *Parrot Linux*. У любого администратора всегда полно работы. Без хорошего набора инструментов его дни – это постоянное испытание на прочность, непрерывная гонка. Однако, существует множество дистрибутивов *Linux*, готовых прийти на помощь. Один из них – *Parrot Linux*. Уверен, он приобретёт серьёзную популярность в 2017-м.

Этот дистрибутив основан на *Debian*, он предлагает огромное количество средств для испытания защищённости систем от несанкционированного доступа. Тут, кроме того, можно найти инструменты из сферы криптографии и компьютерной криминалистики, средства для работы с облачными службами и пакеты для обеспечения анонимности. Есть здесь и кое-что для разработчиков, и даже программы для организации времени. Всё это (на самом деле, там – море инструментов) работает на базе стабильной, проверенной временем системы. В результате получился дистрибутив, отлично подходящий для специалистов по информационной безопасности и сетевых администраторов.

Лучший легковесный дистрибутив *LXLE*. *LXLE* сочетает в себе достойные возможности и скромные системные требования. Другими словами, это дистрибутив, который занимает мало места, но позволяет полноценно работать на компьютере. В *LXLE* можно найти всё необходимое, характерное для релизов *Linux*, рассчитанных на настольные ПК. Система вполне подойдет для дома, под её управлением смогут работать не самые новые компьютеры (не говоря уже о вполне актуальных системах).

*LXLE* основана на *Ubuntu* 16.04 (как результат – долговременная техподдержка обеспечена), здесь применяется менеджер рабочего стола *LXDE*, который, за более чем десять лет существования, знаком многим, да и устроен несложно.

После установки *LXDE* у под рукой окажется множество стандартных средств, вроде *LibreOffice* и *Gimp*. Единственно, надо будет самостоятельно установить современный браузер.

Лучший корпоративный серверный дистрибутив *RHEL*. По данным *Gartner* [3], *RHEL* принадлежит 67% рынка *Linux*-дистрибутивов для крупных организаций, при этом подписка на *RHEL* приносит компании *Red Hat* около 75% доходов. У такого положения дел много причин. Так, *Red Hat* предлагает корпоративным клиентам именно то, что им нужно, но, кроме этого, компания прикладывает огромные усилия к развитию множества проектов с открытым исходным кодом.

*Red Hat* знает – что такое *Linux*, и что такое – корпоративный сектор. *Red Hat* доверяет немало компаний из списка *Fortune* 500 (например, *ING*, *Sprint*, *Bayer Business Services*, *Atos*, *Amadeus*, *Etrade*). Дистрибутив *RHEL* вывел множество разработчиков на новый уровень в областях безопасности, интеграции, управления, в сфере работы с облачными системами.

## 1.5 Нынешнее состояние и тенденции развития

Важно понимать, что организации выбирают *Linux* из-за фактов, а не из-за таблиц сравнения с другими ОС. Возвращаясь к теме фактов о *Linux*, следует сказать, что *Linux* действительно является надежной, гибкой и высокоэффективной ОС. Вот характерный пример применения: инженеры, проводящие многие часы за клавиатурой, переходят с *Windows* на *Linux*, раздраженные постоянной необходимостью перезагрузки. Интернет-провайдеры (*ISP*) переходят с *Windows* на *Linux*, из-за лучшей управляемости последнего.

*Windows* с другой стороны, традиционно держала пальму первенства, когда требовалась простота использования, легкость установки, прогнозируемость обслуживания, и количество приложений.

Область распространения *Linux* огромна, гораздо больше чем у всех других операционных систем. Кроме того, что *Linux* прекрасно работает на обычных домашних и рабочих компьютерах и серверах, существуют адаптации *Linux* к большинству современных процессоров, что позволяет использовать системы с ядром *Linux* в сетевом оборудовании, домашней «умной» технике, роботах, мобильных телефонах, различных портативных устройствах и другом оборудовании, поддерживающем программируемые операции.

В конечном счёте столь широкий круг поддерживаемых устройств означает превосходную переносимость программ. Например, одно и то же приложение зачастую можно запустить с минимальными усилиями и на обычном компьютере, и на мобильном телефоне на базе *Linux*. Для примера: *Windows* и её младший брат *Windows Mobile* являются полностью несовместимыми платформами.

Сейчас *Linux* лучше, чем *Windows* справляется с установкой *plug-and-play* устройств (с простым включением в сеть). Рабочий стол *Linux* можно настроить, чтобы он выглядел не только как *Windows*, но и можно запускать пакеты приложений, которые по функциональности эквивалентны *Microsoft Office*. Реализация новых стандартов и протоколов происходит раньше в *Linux*. Это из-за того, что исходный код легко доступен, «заплаты», для дефектов в аппаратуре, для *Linux* иногда выходят в тот же день.

Современные тенденции в развитии ОС.

На основе опыта использования многих современных ОС, можно выделить следующие основные тенденции в их развитии.

- Графические оболочки. Любая современная ОС имеет графический пользовательский интерфейс, причем (по вполне понятным причинам острой конкуренции между фирмами-разработчиками) графические оболочки для всех ОС примерно одинаковы по возможностям. Подчас пользователю трудно сориентироваться, в какой именно ОС он работает, хотя для конечных пользователей (непрограммистов), по видимому, такая унификация удобна.
- Поддержка новых сетевых технологий и *Web*-технологий. Сети и Интернет активно развиваются. Появляются новые стандарты и протоколы – *IPv6*, *HTML 5* (для облачных вычислений) и т.д. Современные ОС развиваются в направлении поддержки всех новых сетевых технологий.
- Усиленное внимание к механизмам безопасности и защиты. Во многом благодаря инициативе *Trustworthy Computing*, начатой фирмой

*Microsoft* в 2002 г., а также ввиду все усиливающейся киберпреступности, все современные ОС уделяют повышенное внимание безопасности: при просмотре веб-страниц браузеры выполняют их проверку на отсутствие *phishing*; загрузки и инсталляции программ из сети выполняются только с явного согласия пользователя и т.д.

- Поддержка многопоточности и многоядерных процессоров. Ввиду широкого распространения многоядерных процессоров, все современные ОС имеют библиотеки программ, поддерживающие эту возможность аппаратуры. Именно благодаря многоядерной архитектуре, становится реально возможным параллельное выполнение потоков (*threads*).
- Поддержка распределенных и параллельных вычислений. Современные ОС имеют в своем составе высокоуровневые библиотеки, позволяющие разрабатывать параллельные алгоритмы решения задач – например, поддерживающие стандарты параллелизма *OpenMP* и *MPI*.
- Виртуализация ресурсов и аппаратуры. Современные ОС имеют в своем составе средства виртуализации, позволяющие выполнять приложения для других платформ в изолированных виртуальных машинах, в которые могут быть инсталлированы другие операционные системы.
- Развитие файловых систем с целью защиты информации и значительного увеличения размера файлов (для мультимедиа). Современные требования обработки мультимедийной информации приводят к тому, что старые файловые системы (например, *FAT*) оказываются недостаточными для хранения мультимедийных файлов. Например, максимальный размер файла в системе *FAT* – 4 гигабайта – легко может быть превышен при переписи на компьютер цифровой видеопленки длительностью 10-15 минут. Поэтому разрабатываются новые файловые системы, допускающие хранение очень больших файлов, например, система *ZFS* в ОС *Solaris*. Другим требованием является обеспечение конфиденциальности информации, которое приводит к необходимости реализации в файловых системах возможности шифрования (которая реализована, например, в файловой системе *ZFS*).

Благодаря открытости и гибкости *Linux* стала идеальной операционной системой для лабораторий и других исследовательских учреждений, находящихся на переднем крае революционных технологических изменений. Исследования, проводимые в *IBM*, охватывают все области информационных технологий – от физики и когнитологии до передовых прикладных исследований и т.д. Однако ученые корпорации *IBM* также вовлечены в

решение чисто научных проблем. В *IBM*, так же как и во многих других местах, для этого часто используется *Linux*.

*Linux* легко кластеризируется и настраивается для проведения специфических экспериментов, моделирования, выполнения вычислений или тестов. Кроме того, ученые получают в свое распоряжение огромный объем бесплатных программных средств, для работы с которыми и был создан *Linux*. Потенциал и перспективы эры компьютеров, в которой мы живем, еще в значительной мере не исчерпаны, несмотря на появление в настоящее время таких выдающихся технологий, как *Grid*-вычисления, приложения для беспроводной передачи голоса, искусственный интеллект и квантовые вычисления. Надежность, открытость и гибкость *Linux* означают, что эта система будет оставаться на переднем плане разработок в течение многих лет.



## ГЛАВА 2

### Рождения $Z'$ - бозонов в протон-протонных столкновениях с учетом эффектов $Z - Z'$ смешивания

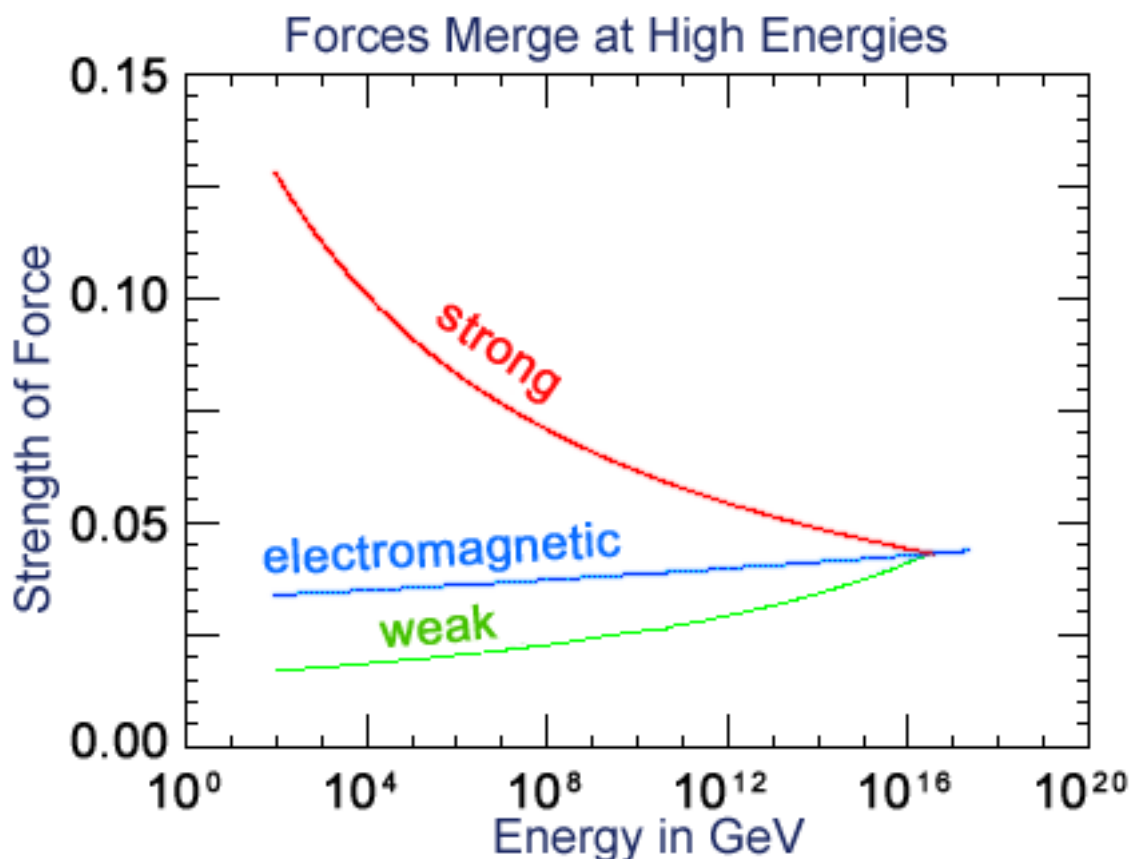
#### 2.1 Новая физика

Несмотря на впечатляющий успех в описании экспериментов, Стандартная модель не может считаться окончательной теорией элементарных частиц. У нее есть свои трудности. Физики уверены, что она должна быть частью некоторой более глубокой теории строения микромира, той частью, которая видна в экспериментах на коллайдерах при энергиях ниже примерно 1 ТэВ. Главная задача Большого адронного коллайдера — получить хотя бы первые намеки на то, что это за более глубокая теория.

Теоретики разработали большое число кандидатов на такую теорию. Все они, естественно, включают какие-то элементы, которые отсутствуют в Стандартной модели. Часто такие теории коллективно называют «Новая физика» или «За пределами Стандартной модели». На этой странице перечислены некоторые из активно изучаемых вариантов Новой физики [5].

Суперсимметрия — это гипотетическая симметрия между фермионами и бозонами. Теории, использующие эту идею, оказываются удивительно мощными, и потому именно с суперсимметрией многие связывают надежды на открытие физики за пределами Стандартной модели. Однако до сих пор не было получено ни одного убедительного доказательства в пользу того, что суперсимметрия реализуется в нашем мире. Ее поиск является одной из главных задач Большого адронного коллайдера. Константы связи трех взаимодействий частиц в микромире сходятся к одному значению, если имеющиеся сейчас данные экстраполировать в область очень высоких энергий. Это совпадение считается неслучайным и воспринимается физиками как намек на то, что все три взаимодействия при больших энергиях объединяются в одно.

В XIX веке физики обнаружили, что электричество и магнетизм — это две стороны одной медали, электромагнитного взаимодействия. Век спустя, при создании Стандартной модели, электромагнетизм и слабые ядерные силы были объединены в рамках единого электрослабого взаимодействия. (Точнее говоря, внутри электрослабого взаимодействия имеются по-прежнему две разные силы, а электромагнитное и слабое взаимодействия возникают как комбинации этих сил). Каждое такое объединение упрощало



**Рисунок 2.1** — Константы связи трех типов взаимодействий

теорию, уменьшало количество введенных в нее «сущностей», переводило наше понимание микромира на новый уровень.

Сейчас физики имеют сразу несколько причин подозревать, что при очень высоких энергиях происходит объединение электрослабого и сильного взаимодействий (рисунок 2.1). Модели, использующие эту идею (так называемые Теории великого объединения) разрабатываются уже давно. В идеале хотелось бы, чтобы такая теория естественным образом объясняла, почему фундаментальных взаимодействий именно столько и именно с такими свойствами, а также имела четкие предсказания, доступные проверке в современных экспериментах.

При энергиях элементарных частиц, доступных на ускорителях, гравитация по-прежнему остается исключительно слабой, так что заметить ее проявления не удастся. Однако ее сила растет с ростом энергии, и при энергиях столкновения порядка планковской она станет столь же важной, как и другие взаимодействия. В этом случае в полный рост встает исключительно сложный вопрос о том, как включить гравитацию в квантовое описание микромира. Поскольку гравитация в современной физике счи-

тается проявлением кривизны пространства-времени, успешная теория с сильной гравитацией должна описывать в рамках единого формализма не только все взаимодействия и всё вещество, но и структуру пространства-времени.

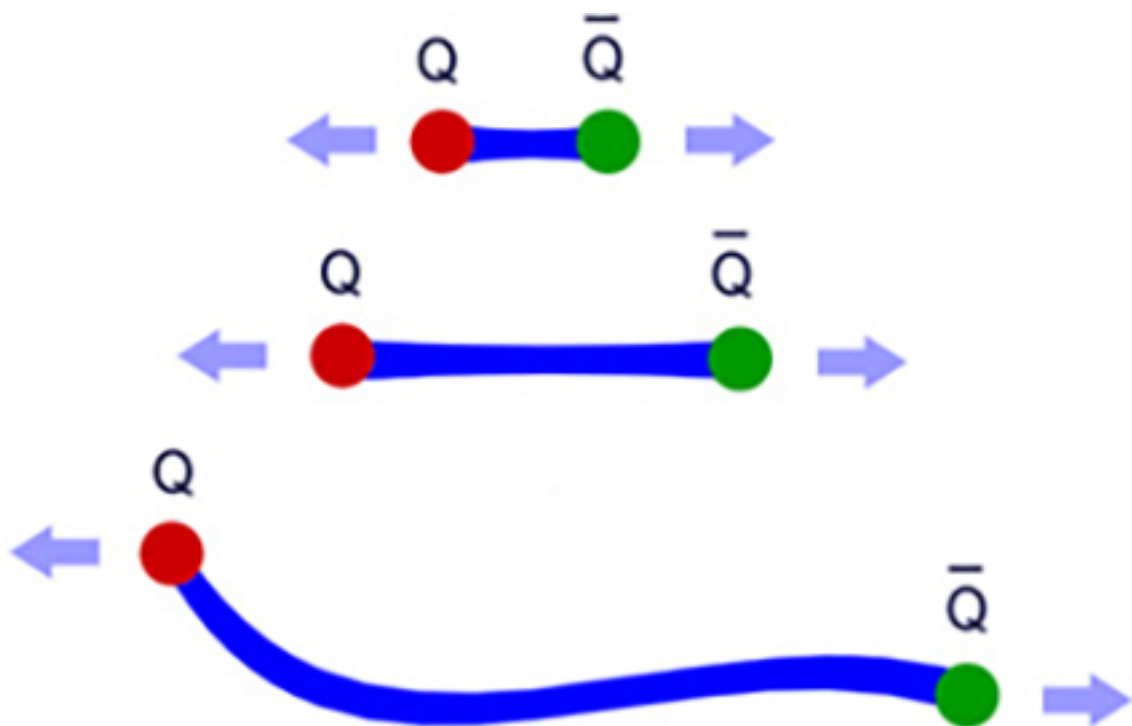
Одним из наиболее привлекательных путей решения этого вопроса является теория суперструн и ее дальнейшее развитие в виде теории бран и М-теории. В этих теориях считается, что фундаментальными объектами, существующими в многомерной вселенной, являются не точечные частицы, а протяженные объекты – струны, мембраны и еще более многомерные образования. В этой теории были получены впечатляющие успехи при высоких энергиях, однако при попытке вывести свойства нашего низкоэнергетического мира из теории суперструн возникает обескураживающая неопределенность предсказаний.

Долгое время казалось, что проверка предсказаний теории суперструн лежит далеко за пределами возможностей человечества, поскольку речь идет об энергиях, на 15 порядков превышающих энергии современных ускорителей. Однако примерно 10 лет назад возникло новое направление развития теории, в котором гравитация становится сильной на энергиях порядка 1 ТэВ. Такая возможность возникает в том случае, если наш мир более чем трехмерный и если при этом новые дополнительные пространственные размерности достаточно протяженны: либо они бесконечны, либо свернуты в многомерные петельки размером много больше ядерного масштаба.

В этом случае на *LHC* следует ожидать целый ряд совершенно замечательных эффектов, отсутствующих в Стандартной модели, например, рождение гравитонов, которые будут улетать из нашего мира в дополнительные измерения, и микроскопических черных дыр, тут же испаряющихся с испусканием множества обычных частиц. Будут также наблюдаться сильные отклонения от предсказаний Стандартной модели в столкновении обычных частиц. Стоит, впрочем, подчеркнуть, что пока нет никаких экспериментальных подтверждений того, что эта красивая гипотеза имеет отношение к нашему миру.

Все три перечисленные выше направления «Новой физики» опираются на глубокие теоретические гипотезы об устройстве нашего мира (суперсимметрия, единство сил, квантово-гравитационная вселенная). Однако кроме этих направлений теоретики также рассматривают разнообразные теории «статусом ниже». В этих теориях просто отмечается, что текущие экспериментальные данные не запрещают те или иные экзотические объекты или явления, и разрабатываются их следствия. Вот несколько примеров таких моделей разной степени экзотичности.

Неминимальные хиггсовские модели. Поскольку хиггсовские бозоны —



**Рисунок 2.2** — Кварки

единственные частицы Стандартной модели, до сих пор не открытые экспериментально, теоретики изучают самые разные варианты устройства этого сектора теории. Новые поколения фермионов. Можно предположить, что кроме трех известных поколений кварков и лептонов существуют и другие поколения. Частицы из этих поколений должны быть очень тяжелыми, иначе бы их уже давно открыли в эксперименте.

Новые короткодействующие силы. В таких моделях предполагается, что в нашем мире есть и иные силовые взаимодействия, отличные от сильных, слабых и электромагнитных, но они настолько короткодействующие, что до сих пор никак не проявлялись в эксперименте. На Большом адронном коллайдере благодаря его рекордной энергии удастся «прощупать» взаимодействия частиц на исключительно малых расстояниях (менее  $10^{-19}$  метра), а значит, появляется шанс эти взаимодействия обнаружить. Они могут проявляться либо как рождение и распад частицы-переносчика новых сил (такие гипотетические частицы обозначают  $Z'$ ), либо как усиленное рассеяние частиц на большие углы.

Лептокварки. В Стандартной модели и в подавляющем большинстве теорий Новой физики кварки и лептоны взаимодействуют друг с другом опосредованно, путем обмена квантами силовых полей. Однако можно представить себе возможность того, что кварки и лептоны исходно явля-

лись фермионами одного типа и лишь потом расщепились на два разных сорта. В таком случае должны существовать новые тяжелые частицы — лептокварки, которые распадаются прямо на кварк и лептон. Подобные частицы встречаются в теориях Великого объединения.

Квирки. Одним из очень необычных и любопытных вариантов новых сил является гипотеза квирков (*quirks*). Эта модель построена по типу обычного сильного взаимодействия: в ней предполагается, что существует новое силовое поле с конфайнментом и новые частицы, его чувствующие. Если частицы очень тяжелые, то между ними будут натягиваться длинные, даже макроскопические силовые струны, которые не смогут порваться (рис. 2.2).

Слабое взаимодействие — короткодействующее фундаментальное взаимодействие между элементарными частицами, ответственное за бета-распад атомных ядер и медленные распады частиц. Слабое взаимодействие значительно слабее сильного и электромагнитного, но гораздо сильнее гравитационного. В слабом взаимодействии участвуют все фундаментальные фермионы (кварки и лептоны) и все адроны. Единственными частицами, которые участвуют только в слабом взаимодействии являются три типа нейтрино  $\nu_e, \nu_\mu, \nu_\tau$  и их античастицы антинейтрино  $\bar{\nu}_e$ , антинейтрино  $\bar{\nu}_\mu$ , антинейтрино  $\bar{\nu}_\tau$ . В нем не участвуют переносчики сильного, электромагнитного и гравитационного взаимодействий — глюон, фотон и гравитон. В процессе слабого взаимодействия частицы обмениваются переносчиками слабого взаимодействия промежуточными (фундаментальными) бозонами: имеющими электрический заряд  $W^\pm$  и нейтральным  $Z$ . Эти бозоны, в отличие от переносчиков остальных фундаментальных сил безмассовых глюона, фотона и гравитона, имеют огромные массы  $m_W = 80.4 \text{ ГэВ}/c^2$  и  $m_Z = 91.2 \text{ ГэВ}/c^2$  (примерно как у атомов циркония или ниобия), что приводит к очень малому радиусу действия слабых сил  $\approx 10^{-18}$  см (что на три порядка меньше радиуса сильного взаимодействия) и очень низкой по сравнению с сильными и электромагнитными процессами вероятности (скорости) слабых процессов.

Несмотря на малую величину и короткодействие слабые силы играют очень важную роль в природе. Так без них погасло бы Солнце, так как внутри него остановился бы процесс превращения 4 протонов в ядро гелия-4, являющийся основным источником энергии Солнца.

Слабое взаимодействие выделяется тем, что в нём не соблюдается ряд запретов, присущих сильному и электромагнитному взаимодействиям. Так в слабых процессах кварки одного типа (аромата) превращаются в кварки других ароматов [9].

Особенности слабого взаимодействия

- Их слабость (медленность), выражающаяся в том, что вероятность этих процессов на много порядков меньше вероятностей сильных и электромагнитных процессов.
- Малый радиус взаимодействия — как минимум на два порядка меньший, чем радиус сильного взаимодействия. Ни в одном из слабых процессов не удалось до 1982 г. обнаружить каких-либо отклонений от точечного четырех- фермионного взаимодействия.
- Сильное, максимально возможное несохранение пространственной и зарядовой четностей. Так, в заряженные токи входят только левые компоненты спиноров, описывающих частицы, и только правые компоненты спиноров, описывающих античастицы.
- Несохранение  $CP$ -четности.
- Несохранение ароматов (странности, чарма и т. д.).
- То обстоятельство, что только в слабых взаимодействиях принимают участие нейтрино.

Тем поразительней, что, несмотря на столь резкие отличия, слабые и электромагнитные взаимодействия представляют собой, по-видимому, проявление одного и того же взаимодействия, которое в последние годы получило название электрослабого.

Согласно электрослабой теории слабые взаимодействия заряженных токов обусловлены обменами  $W$ -бозонами, а нейтральных —  $Z$ -бозонами, подобно тому как взаимодействие электромагнитных токов обусловлено обменом фотонами. При этом слабость и малый радиус слабого взаимодействия объясняются тем, что, в отличие от фотонов,  $W$  и  $Z$ -бозоны — очень тяжелые частицы. Остальные особенности слабого взаимодействия прямо заложены в предположении о форме исходных фермионных токов теории. Так что в электрослабой теории удивляться надо не тому, что слабое взаимодействие зеркально-асимметрично, а тому, что электромагнитное — зеркально-симметричное.

Слабое взаимодействие переносится массивными  $W^\pm$ - и  $Z$ -бозонами. Обмен заряженными  $W^+$  и  $W^-$ -бозонами приводит к изменению электрического заряда взаимодействующих фермионов. Эти процессы происходят за счет заряженных токов.

## 2.2 Инструменты имитационного моделирования

Для исследования отклика детектора на различные физические процессы, созданы программы, позволяющие перевести моделированное на

уровне частиц событие взаимодействия протонов при соударении в формат представления данных детекторов установки *ATLAS*. Алгоритмы моделирования интегрированы в программную оболочку эксперимента *ATLAS*, именуемую *Athena*, использующую программный пакет *GEANT4*.

Генератор события создает набор частиц, который направляется в программу быстрого или полного моделирования детектора. Генераторы событий встроены в *Athena*. Используется большое число других, поддерживаемых авторами, генераторов, которые имеют блоки связи для использования в *Athena*. Основной массив модельных событий создан с помощью генераторов *PYTHIA* [7], включая его версию *PYTHIAB*, предназначенную в *ATLAS* для моделирования событий с рождением *B*-адронов.

*PYTHIA* - это программного пакета для визуализации результатов моделирования процессов столкновения частиц при высоких энергиях осуществляющего генерацию методом Монте-Карло физических событий.

Программы *PYTHIA* интенсивно используются для генерации событий в физике высоких энергий при описании процессов множественного рождения в столкновениях элементарных частиц. В частности задачи, что включает решаемые жесткие с взаимодействия помощью данного в столкновениях  $e^+e^-$ ,  $pp$  и  $ep$ , а также некоторые другие случаи. Программа предназначена для генерации генератора полных событий, т.е. дают более детальную картину, чем мы наблюдаем в эксперименте, в рамках нашего понимания фундаментальной физики процессов. Обсуждаемые здесь программы Монте-Карло построены как ведомые системы, т.е. пользователь должен написать основную программу. Из нее различные программы вызываются для выполнения частных задач, после чего управление снова передается основной программе. Некоторые из этих задач могут быть весьма тривиальными, и достаточно высокоуровневые программы могут производить большое число вызовов подпрограмм.

Генераторы общего назначения создают событие как целое. Они используют много параметров, часть из которых относится к фундаментальным параметрам, такие как константы связи квантовой хромодинамики (КХД) и электрослабой теории, часть относится к моделям, описывающим взаимодействия на больших расстояниях, с малыми передачами импульса, т.н. «мягкой» КХД, и к электрослабым процессам.

Для корректного моделирования процессов рождения и распада частиц необходимо учитывать условия проведения эксперимента. Это условия рождения изучаемых частиц на ускорителе при соответствующих энергиях сталкивающихся пучков, полные цепочки распадов частиц до уровня «стабильных частиц», регистрируемых детектором. Для решения этих задач применяются генераторы событий, использующие метод Монте-Карло.

Генератор *PYTHIA* является широко используемой в физике высоких энергий программой моделирования столкновений различных частиц в широком диапазоне энергий. Этот генератор учитывает процессы фрагментации кварков в адроны и разыгрывает сложные цепочки адронных распадов. Стартуя с заданного пользователем процесса, (столкновение двух протонов с рождением  $Z$ -бозона и т.п.) программа случайным образом (с учетом законов сохранения и, по возможности, теоретически известной структуры взаимодействия) разыгрывает конфигурацию конечных партонов, а затем моделирует т.н. процесс адронизации - процесс превращения ненаблюдаемых кварков и глюонов в реальные стабильные и нестабильные частицы с последующим распадом нестабильных частиц. На выходе программа выдает список всех частиц, родившихся в результате столкновения заданных первичных частиц, значения их компонент импульса и энергии. Кроме того, имеется возможность проследить последовательность рождений и распадов от первичного взаимодействия до рождения данной частицы. В качестве входных параметров программы используются описания сталкивающихся частиц, их энергий и тип моделируемого процесса (например, рождение  $Z$ -бозона). Существующие версии пакета *PYTHIA* написаны на языке программирования *FORTRAN*. Результаты генерации – характеристики вторичных частиц – записываются в файл, что позволяет в дальнейшем проводить статистическую обработку событий.

### 2.3 Распады промежуточных бозонов. Определение массы $Z'$ -бозона

В физических программах экспериментов на современных дронных (*LHC*) и планируемых на электрон-позитронных (*ILC*, *CLIC*) коллайдерах вопросу поиска «новой» физики, выходящей за рамки Стандартной модели (СМ), традиционно уделяется большое внимание. К числу подобных теоретических построений, являющихся обобщением СМ, относятся модели с расширенным к либровочным сектором, такие как лево-правосимметричные модели (*LR*), альтернативные лево-правосимметричные модели (*ALR*),  $E_6$ -модели и др. [8]. Их исследование (теоретическое и экспериментальное) представляет значительный интерес. Эти модели являются одними из простейших расширений СМ, характеризующихся элементарной структурой хиггсовского сектора. Общим для данных моделей является то, что они предсказывают новые физические объекты и явления на масштабе энергий  $O(1 \text{ ТэВ})$ , связанные, например, с наличием тяжелых нейтральных ( $Z'$ ) калибровочных бозонов, обусловленных дополнительными



калибровочными симметриями  $U(1)'$ .

Достижение порога рождения  $Z'$ -бозона явилось бы прямым доказательством проявления «новой» физики. Однако в данном случае интервал поиска масс  $Z'$  ограничен максимальной энергией коллайдера, на котором проводятся эксперименты. Значительно более широкий интервал масс можно исследовать с помощью пропагаторных эффектов. В этом случае ведется поиск отклонений различных наблюдаемых от соответствующих предсказаний СМ. Если экспериментальные данные при достигнутом уровне точности согласуются с СМ, т. е. отклонений от предсказаний СМ нет, то эту экспериментальную информацию можно использовать для получения ограничений на динамические параметры и массы  $Z'$ -бозонов.

Потенциальные возможности  $e^+e^-$ -коллайдеров для прямого рождения новых калибровочных бозонов гораздо скромнее по сравнению с адронными машинами из-за более низких энергий пучков. Кроме того, современные ограничения на массы  $Z'$ -бозонов для большинства моделей превосходят планируемую энергию электрон-позитронного коллайдера  $ILC$ ,  $\sqrt{s} \ll M_{Z'}$ . Тем не менее основным достоинством этих машин является возможность проведения экспериментов по измерению наблюдаемых величин с высокой степенью точности и получения однозначной информации о косвенных (виртуальных) эффектах новых  $Z'$ -бозонов, а также эффектах бозонного  $Z$ - $Z'$ -смешивания. Последние, в моделях с расширенным калибровочным сектором, зависят от структуры хиггсовского сектора модели. Тем самым экспериментальное исследование процессов рождения пар  $W^\pm$ -бозонов может не только пролить свет на возможное существование «новой» физики, но и дать косвенные указания на хиггсовскую природу, а также установить структуру модели.

На основе данных, полученных из низкоэнергетических экспериментов по нейтральным токам, результатов на  $e^+e^-$ -коллайдерах  $LEP$  и  $SLC$  [8], а также недавно выполненных экспериментов по поиску прямого адронного рождения  $Z'$ -бозонов в процессе Дрелла-Яна.

$$pp \rightarrow Z' \rightarrow l^+l^- + X$$

( $l = e, \mu$ ) на коллайдере  $LHC$  при энергии  $\sqrt{s} = 7$  и 8 ТэВ с интегральной светимостью соответственно  $L_{int} = 5$  и 20  $\text{fb}^{-1}$  [8] можно заключить, что для большинства расширенных калибровочных моделей граничные значения для масс дополнительных  $Z'$ -бозонов находятся в интервале  $\sim 2,5\text{--}3,0$  ТэВ (в зависимости от модели), а современный масштаб ограничений на угол смешивания составляет  $\mathcal{O}(\varphi) \sim 10^{-2}\text{--}10^{-3}$  рад. При этом наиболее точная информация об угле смешивания была получена преимущественно из экспериментов на электрон-позитронных коллайдерах  $LEP1$  [12] и  $SLC$

по измерению резонансных наблюдаемых физических величин при энергии начальных состояний, равной массе стандартного  $Z$ -бозона,  $\sqrt{s} = M_Z$ , в процессах

$$e^+e^- \rightarrow f\bar{f}$$

где конечными фермионными состояниями  $f$  были заряженные лептоны и кварки [11]. Высокая точность, достигнутая в экспериментах на коллайдерах  $LEP1$  и  $SLC$ , объясняется прежде всего возможностью набора большого объема данных в резонансной области энергии.

Кроме того, эта информация дополнялась данными, полученными на коллайдере тэватрон, по точному измерению массы  $M_W$ , на основе которых определялся параметр бозонного  $Z-Z'$ -смешивания с использованием соотношения между массами нейтральных и заряженных калибровочных бозонов,  $M_Z = M_W/(\sqrt{p_0} \cos \theta_W)$ , имеющего место в расширенных моделях. Очевидно также, что эти данные будут дополнены новой информацией, которая в ближайшем будущем будет получена в экспериментах на коллайдере  $LHC$  при энергии 13 и 14 ТэВ. Вместе с тем из этих данных нельзя сделать однозначный вывод о природе «новой» физики, который мог бы вызвать отклонение наблюдаемых величин от их поведения, предсказываемого СМ. Дело в том, что параметр  $p$ , который содержится в выражениях для векторных и аксиально-векторных констант связи фермионов с учетом петлевых поправок, зависит, в частности, от структуры хиггсовского сектора модели, которая изначально неизвестна. Кроме того, новые тяжелые фермионы и скалярные частицы, предсказываемые моделями с расширенным калибровочным сектором, могут давать вклад в параметр  $p$  на петлевом уровне. Все эти неопределенности приводят к появлению систематических (теоретических) погрешностей, которые могут быть весьма существенными при измерении параметра  $p$  и, в конечном счете, могут повлиять на точность определения параметра  $Z-Z'$ -смешивания.

Процессы парного рождения заряженных  $W^\pm$ -бозонов в адронных столкновениях на  $LHC$

$$pp \rightarrow W^+W^- + X$$

электрон-позитронной аннигиляции на  $LEP2$  и в большей степени на  $ILC$

$$e^+e^- \rightarrow W^+W^-$$

Являются весьма эффективным инструментом поиска эффектов  $Z-Z'$ -смешивания при высоких энергиях и, таким образом, играют роль основного поставщика информации об угле  $Z-Z'$ -смешивания [8]. С теоретической точки зрения процессы парного рождения заряженных калибровочных бозонов в адронных и электронпозитронных столкновениях интересны тем,

что их сечения пропорциональны углу  $Z-Z'$ -смешивания, который, как отмечалось выше, в расширенных калибровочных моделях зависит от структуры хиггсовского сектора [13].

Прямой поиск тяжелых резонансов в процессе  $p\bar{p} \rightarrow W^+W^- + X$  осуществлялся экспериментальными группами  $CDF$  и  $D0$  на коллайдере тэватрон. Коллаборация  $D0$  исследовала возможность рождения резонанса в канале его дибозонного распада, используя чисто лептонные  $lv'l'v'$  и полулептонные  $vjj$  моды. Здесь  $l = e, \mu$ ;  $jj$  — две адронные струи. Коллаборация  $CDF$  также осуществляла поиск тяжелых резонансов в канале их распада в пару заряженных калибровочных бозонов  $W^+W^-$  с последующим распадом в полулептонные  $evjj$  конечные состояния. Обе коллаборации установили ограничения на массы тяжелых резонансов, таких как новые нейтральные  $Z'$ - и заряженные калибровочные  $W^\pm$ -бозоны, гравитоны Рэндалл-Сандрума. Кроме того, в настоящее время поиск тяжелых резонансов на  $LHC$  в  $WW$ -канале интенсивно ведется коллаборациями  $ATLAS$  и  $CMS$ . В частности, уже получена экспериментальная информация о процессе в лептонном канале  $lv'l'v'$  при энергии коллайдера 7 ТэВ и интегральной светимости  $4,7 \text{ фб}^{-1}$  [10].

Из анализа экспериментальных данных по измерению процесса электрон-позитронной аннигиляции на коллайдере  $LEP2$  были впервые получены прямые ограничения на угол  $Z-Z'$ -смешивания. Точность измерения угла смешивания оказалась не очень высокой,  $|\phi|$  5–10 %, так как сам коллайдер работал в интервале энергий, незначительно превышающем порог реакции,  $\sqrt{s} \gg 2M_W$ . Как было установлено ранее, чувствительность процесса электрон-позитронной аннигиляции к эффектам «новой» физики значительно усиливается при высоких энергиях,  $\sqrt{s} \gg 2M_W$ , где важную роль играет механизм калибровочного сокращения. Дело в том, что вклад  $Z'$ -бозона в сечение процесса нарушает механизм калибровочного сокращения, играющий важную роль в СМ. Действие механизма калибровочного сокращения состоит в том, что он обеспечивает «правильное» поведение сечения процесса электрон-позитронной аннигиляции с ростом энергии, которое не нарушает унитарный предел, несмотря на быстро растущие с энергией отдельные вклады в сечение. Вместе с тем эффекты, индуцированные появлением дополнительного калибровочного бозона, нарушают механизм калибровочного сокращения в энергетическом интервале  $2M_W \ll \sqrt{s} \ll M_{Z'}$ , что проявляется в виде «разбалансировки» отдельных вкладов в сечение и, как следствие, в возникновении существенно иной по сравнению со СМ энергетической зависимости сечений. Этим обусловлено действие так называемого механизма усиления эффектов «новой» физики в процессе электрон-позитронной аннигиляции.

Именно в силу этого обстоятельства линейный коллайдер  $ILC$  является одним из основных инструментариев для поиска эффектов «новой» физики при исследовании процесса электрон-позитронной аннигиляции.

Следует отметить также, что коллаборация  $CDF$  на коллайдере тэватрон одной из первых получила прямые ограничения на угол  $Z-Z'$ -смешивания из обработки данных по измерению процесса адронного рождения  $W^+W^-$ -бозонов. И вновь относительно небольшая энергия установки и низкая светимость не позволили улучшить ограничения, полученные на коллайдере  $LEP2$ , а лишь повторить их [14].

Возможности коллайдера  $LHC$  по обнаружению эффектов  $Z-Z'$ -смешивания в процессе рождения пар заряженных калибровочных  $W^\pm$ -бозонов с их последующим распадом по чисто лептонному каналу  $lv'l'v'$ . Несмотря на очевидное достоинство данного канала, связанное с подавленностью фона, особенно при больших инвариантных массах  $W^\pm$ -бозонов, у него имеется заметный недостаток, связанный с присутствием в конечных фермионных состояниях двух нейтрино, что не позволяет восстановить распределение по инвариантной массе бозонных пар из экспериментальных данных. В то же время распад пары  $W^\pm$ -бозонов по полулептонному каналу  $lvjj$  свободен от указанного недостатка. В процессе  $pp \rightarrow Z' \rightarrow WW + X \rightarrow lvjj + X$  существует возможность реконструировать распределение по инвариантной массе  $W^+W^-$ -пары и тем самым исследовать резонансную структуру  $Z'$ -бозона. Еще одним достоинством настоящего полулептонного процесса является то, что он имеет сечение, существенно превосходящее сечение чисто лептонного канала. Вместе с тем полулептонный канал, в отличие от лептонного канала  $lv'l'v'$ , имеет большой КХД-фон, вызванный рождением  $W_{jj}$ -, а также  $Z_{jj}$ -состояний [15]. В последнем случае предполагается, что  $Z$ -бозон распадается по лептонному каналу, а в процессе детектирования лептонов один из них теряется. Кроме перечисленных выше КХД фоновых процессов имеется еще один, который играет важную роль в оценке всей фоновой составляющей. Это процесс рождения пар  $t\bar{t}$ -кварков. Однако большой КХД-фон может быть редуцирован путем наложения кинематических ограничений на поперечные импульсы заряженных лептонов и адронных струй в резонансном сигнале рождения  $Z'$ -бозонов [8].

## ЗАКЛЮЧЕНИЕ

Целью реферата было исследование преимуществ и недостатков ОС *Linux*. Был поставлен ряд задач, которые необходимо было выполнить, для достижения намеченной цели. Если рассмотреть последовательно каждый пункт, то можно сделать вывод, что цель реферата достигнута: дан развернутый ответ на вопрос, что такое *Linux*. Рассмотрена поэтапно история создания ОС *Linux*. Проанализированы сильные и слабые стороны современных ОС, а также выявлены основные преимущества и недостатки. Сделаны соответствующие выводы о перспективе развития *Linux*.

Во второй главе реферата рассмотрен процесс рождения  $Z'$  бозонов в протон-протонных столкновениях с учётом эффектов  $Z$ - $Z'$  смешивания. Рассмотрены инструменты библиотеки *PYTHIA* для имитационного моделирования процессов взаимодействия элементарных частиц при высоких энергиях. Исследован процесс рождения  $Z'$ -бозонов в процессе  $pp \rightarrow Z' \rightarrow l^+l^- + X$  с учетом эффектов  $Z$ - $Z'$  смешивания.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Назаров, С. В. Современные операционные системы: учебное пособие / С. В. Назаров, А. И. Широков. — Москва : Национальный Открытый Университет «ИНТУИТ», 2012.
2. Основные понятия ОС [Электронный ресурс]. — Режим доступа: <http://technomag.bmstu.ru/doc/48639.html> — Дата доступа: 11.12.2017.
3. Операционные системы Linux [Электронный ресурс]. — Режим доступа: <http://help.ubuntu.ru/wiki/linux> — Дата доступа: 11.12.2017.
4. Linux-2017: самые перспективные дистрибутивы [Электронный ресурс]. — Режим доступа: <https://habrahabr.ru/company/ruvds/blog/320002/> — Дата доступа: 11.12.2017.
5. За пределами Стандартной модели [Электронный ресурс]. — Режим доступа: <https://elementy.ru/LHC/HEP/SM/beyondSM> — Дата доступа: 11.12.2017.
6. Н. В. Красников, В. А. Матвеев. Поиск новой физики на LHC [Электронный ресурс]. — Режим доступа: [http://nuclphys.sinp.msu.ru/ATLAS\\_exp/at03.htm](http://nuclphys.sinp.msu.ru/ATLAS_exp/at03.htm) — Дата доступа: 11.12.2017.
7. Official documentation [Электронный ресурс]. — Режим доступа: <http://home.thep.lu.se/torbjorn/Pythia.html> — Дата доступа: 11.12.2017.
8. Бобовников, И.Д. Эффекты  $Z - Z'$ -смешивания в процессах рождения пары  $W^\pm$ -бозонов на адронных и лептонных коллайдерах высоких энергий / И.Д. Бобовников, А.А. Панков. — Письма в ЭЧАЯ, 2016. Т. 13, №1(199). С.8-35
9. Слабое взаимодействие [Электронный ресурс]. — Режим доступа: <http://nuclphys.sinp.msu.ru/enc/e149.htm> — Дата доступа: 11.12.2017.
10. Osland, P. Probing  $Z - Z'$  mixing with ATLAS and CMS resonant diboson production data at the LHC at  $\sqrt{s} = 13$  TeV / P. Osland,

- A.A. Pankov, A.V. Tsytrinov // Physical Review D. – 2012. – Vol. 86. – P. 12.
11. Andreev, V. V. Constraints on the  $Z - Z'$  mixing angle from data measured for the process  $e^+e^- \rightarrow W^+W^-$  at the LEP2 collider / V.V. Andreev, A.A. Pankov // Phys. At. Nucl. – 2012. – Vol. 75. – P. 76.
  12. ALEPH and DELPHI and L3 and OPAL and SLD Collaborations and LEP Electroweak Working Group and SLD Electroweak Group and SLD Heavy Flavour Group / Schael, S. [et al.] // Precision electroweak measurements on the Zresonance, Phys. Rep. 2006. – P. 427.
  13. Search for massive resonances decaying into  $WW$ ,  $WZ$  or  $ZZ$  bosons in proton-proton collisions at  $\sqrt{s} = 13$  TeV / Sirunyan, A. M. [et al.] // J. High Energy Phys. – 2017. – Vol. 162. – P. 56.
  14. Measurement of  $W^+W^-$ –production in pp collisions at  $\sqrt{s} = 7$  TeV with the ATLAS detector and limits on anomalous  $WWZ$  and  $WW_y$  couplings / Ada, G. [et al.] // Phys. Rev. D. – 2013. – Vol. 88. – P. 29.
  15. Search for new phenomena in the  $WW \rightarrow ll'v'$  final state in  $pp$  collisions at  $\sqrt{s} = 7$  TeV with the ATLAS detector / Ada, G. [et al.] // Physics Letters B. – 2013. – Vol. 3. – P. 878.