Федеральное государственное автономное образовательное учреждение высшего образования
«Московский физико-технический институт (национальный исследовательский университет)»

УТВЕРЖДЕНО
Директор физтех-школы прикладной математики и информатики
А.М. Райгородский

Рабочая программа дисциплины (модуля)

по дисциплине: Основы IT-технологий
по направлению: Прикладная математика и информатика
профиль подготовки: Искусственный интеллект и большие данные
Сетевое обучение
кафедра алгоритмов и технологий программирования
курс: 1
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
2 (весенний) - Дифференцированный зачет
3 (осенний) - Дифференцированный зачет

Аудиторных часов: 120 всего, в том числе:
лекции: 60 час.
семинары: 60 час.
лабораторные занятия: 0 час.

Самостоятельная работа: 164 час.

Всего часов: 288, всего зач. ед.: 8

Программу составил: В.В. Яковлев, канд. физ.-мат. наук, заведующий кафедрой

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 30.08.2022

Аннотация

Курс посвящен базовым вопросам обеспечения и организации качественной промышленной разработки ПО и низкоуровневым аспектам разработки программного обеспечения для UNIX-подобный операционных систем, а также отработки навыков написания программ и их тестирования в предельных ситуациях.

Рассматриваются необходимые инструменты и технологии для организации сборки ПО, тестирования, CI/CD процессов. Большое внимание уделяется базовым принципам построения архитектуры ПО, паттернам и антипаттернам разработки. В рамках курса слушатели выполняют ряд практических заданий по отдельным темам курса, а также реализуют большое сквозное проектное задание.

В рамках данной дисциплины будут немного затронуто программирование на языках ассемблера под архитектуры компьютеров ARM (32 бит) и x86, - в объеме, минимально необходимом для понимания таких аспектов, как работа с памятью, соглашения о вызовах, и способы системных вызовов.

После прохождения тем про язык ассемблера, оставшаяся часть курса будет посвящена изучению системных вызовов для работы с памятью, файлами, процессами. Особое внимание будет уделено механизмам межпроцессных взаимодействий: сигнала, каналам, разделяемой памяти, и сетевому взаимодействию.

1. Цели и задачи

Цель дисциплины

овладение студентами технологических приемов, повсеместно применяемых при разработке программного обеспечения. Познакомить студентов с базовыми принципами организации внутренней организации компьютерных систем, с базовыми принципами организации операционных систем, а также абстракций и интерфейсов, которые предоставляются программисту для взаимодействия с операционной системой.

Задачи дисциплины

приобретение студентами навыков работы в командной строке, инструментами сборки и системами контроля версий;

овладение студентами современными практиками разработки и типовыми шаблонами проектирования.

задача дисциплины заключается в демонстрации базовых принципов на примере операционных систем семейства UNIX и, частично, Windows.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-2 Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-2.1. Применяет знания основных положений и концепций в области программирования, архитектуру языков программирования, основную терминологию и базовые алгоритмы, основные требования
	ОПК-2.2. Анализирует типовые языки программирования, составляет программы
	ОПК-2.3 Применяет на практике опыт решения задач с использованием базовых алгоритмов, анализа типов коммуникаций и интеграции различных типов программного обеспечения

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

шаблоны проектирования программного обеспечения;
основы работы в UNIX-подобных системах;
основы низкоуровнего программирования;
основы машинного кода, языков ассемблера;
различные пути повышения производительности программы;
основы сетевого взаимодействия;
основы устройства сетей.

уметь:

работать с интерфейсом командной строки;
выполнять сборку программ из исходных текстов и их отладку, без использования интегрированных средств разработки;
пользоваться системами контроля версий;
настраивать окружение для непрерывной интеграции разработки проекта;
проектировать программное обеспечение таким образом, чтобы его поддержка осуществлялась коллективом из нескольких разработчиков;
создавать многопоточные и межсетевые программы на языках Си и Ассемблер;
работать в unix-подобных средах;
создавать программы на языках Си и Ассемблер без использования высокоуровневых библиотек.

владеть:

навыками работы с GitLab и GitLab CI
навыками ведения простейших программных проектов в системах контроля версий.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Введение в ОС Linux	3	3		14
2	Системы контроля версий	3	3		10
3	Процесс компиляции	4	4		14
4	Организация процесса сборки	3	3		10
5	Кросс-компиляция и методы отладки	3	3		14
6	Порождающие паттерны проектирования	4	4		10
7	Структурные паттерны	3	3		14
8	Поведенческие паттерны	3	3		14
9	Модели	4	4		14
10	Низкоуровневые конструкции языка Си	16	16		25
11	Архитектура 32 и 64-разрядных систем ARM и x86_64	16	16		25
Итого часов		60	60		160
Общая трудоёмкость		288 час., 8 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 2 (Весенний)

1. Введение в ОС Linux

- работа с командной строкой;
- организация файловой системы.

2. Системы контроля версий

- работа с ответвлениями;
- организация совместной работы.

3. Процесс компиляции

- стадии компиляции;
- промежуточные артефакты сборки и их исследование.

4. Организация процесса сборки

- сборка с помощью сценария Makefile;
- высокоуровневые системы сборки;
- непрерывная интеграция.

5. Кросс-компиляция и методы отладки

- кросс-компиляция для другой архитектуры процессора;
- кросс-компиляция для другой операционной системы;
- отладка программ с использованием средств виртуализации.

6. Порождающие паттерны проектирования

- синглетоны;
- фабричные методы;
- прототипы.

7. Структурные паттерны

- адаптеры;
- связки;
- композиты;
- прокси и декораторы.

8. Поведенческие паттерны

- стратегии;
- интерпретатор, итератор, состояние;
- стратегия «наблюдателя».

9. Модели

- паттерн «модель-контроллер-представление».

Семестр: 3 (Осенний)

10. Низкоуровневые конструкции языка Си

Введение в язык Си. Современный диалект языка Си (стандарт 2011 года). Отличия от C++, размещение данных в памяти, выравнивание данных, структуры и объединения, указатели на функции. Представление целых чисел. Обратный дополнительный код, битовые операции. Знаковые и беззнаковые числа. Undefined Behaviour.

11. Архитектура 32 и 64-разрядных систем ARM и x86_64

Язык ассемблера ARM, базовые инструкции. Стек вызовов и вызов функций на ARM. Представление вещественных чисел IEEE754. Программные прерывания и системные вызовы. Ассемблер x86_64. Архитектура CISC v.s. RISC. gdb и objdump. Соглашения о вызовах x86_64. Выравнивание данных и векторные инструкции SSE/AVX.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система).

6.Перечень рекомендуемой литературы

Основная литература

1. Компьютерные сети: принципы, технологии, протоколы [Текст] / В. Г. Олифер, Н. А. Олифер - СПб.Питер,2016

2. Архитектура компьютера [Текст] : [учеб. пособие для вузов] / Э. Таненбаум, Т. Остин ; [пер. с англ. Е. Матвеев] .— 6-е изд. — СПб. : Питер, 2014 .— 816 с

Дополнительная литература

1. Операционная система UNIX [Текст] : учеб. пособие для вузов / А. М. Робачевский .— СПб. : БХВ-Петербург, 2000, 2002, 2003 .— 656 с.

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

http://docs.oracle.com/javase/specs/jls/se8/html/index.html - The Java Language Specification.
https://google-styleguide.googlecode.com/svn/trunk/javaguide.html — Google Java Style Guide.

Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

1. https://git-scm.com/book/ru/v2
2. https://www.gnu.org/software/bash/manual/bash.html

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система).
Операционная система Linux с правами системного администратора.
Возможно использование системы через средства виртуализации.
Стандартные средства разработки, входящие в состав ОС Linux.

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя:

– проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств;

– подготовку к практическим занятиям, выполнение двух индивидуальных домашних заданий.

Промежуточный контроль знаний проводится в виде письменных опросов по теории, а также студенту в ходе освоения курса необходимо выполнить две домашние индивидуальные работы с их последующей защитой:

Зачет выставляется на основе работы на семинаре и выполнения домашних работ, либо, в случае пересдачи комиссии, выполнения задания и его защиты комиссии.

Оценка за зачет выставляется из соотношения: 30% за теоретическую часть, и 70% - за практическую.

Оценка за теоретическую часть - это среднее арифметическое, полученное из оценок на письменных контрольных работах, по итогам каждого из разделов дисциплины.

Оценка за практическую часть - это оценка за выполнение семинарских и домашних заданий, с учетом сроков сдачи.

Внимание: неудовлетворительная оценка за каждую из частей является БЛОКИРУЮЩЕЙ, то есть, в случае неудовлетворительной оценки за теоретическую либо практическую часть, итоговая оценка - неудовлетворительно.

Внимание: выполнение и сдача задач, разбираемых на семинарских занятиях, и задач домашнего задания, помеченных как “обязательные” (как правило, по одной задаче в неделю) является обязательным условием получения положительной оценки.

Все промежуточные расчеты оценки выполняются с точностью до второго знака после точки, итоговая оценка выставляется по 10-балльной шкале, с округлением по стандартным арифметическим правилам.

Дополнительная литература

Брайант Р.Э., О’Халларон Д.Р. Компьютерные системы: архитектура и программирование. СПб.: БХВ, 2005. 1186 с.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Прикладная математика и информатика
профиль подготовки: Искусственный интеллект и большие данные
Сетевое обучение
кафедра алгоритмов и технологий программирования
курс: 1
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
2 (весенний) - Дифференцированный зачет
3 (осенний) - Дифференцированный зачет

Разработчик: В.В. Яковлев, канд. физ.-мат. наук, заведующий кафедрой

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-2 Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-2.1. Применяет знания основных положений и концепций в области программирования, архитектуру языков программирования, основную терминологию и базовые алгоритмы, основные требования
	ОПК-2.2. Анализирует типовые языки программирования, составляет программы
	ОПК-2.3 Применяет на практике опыт решения задач с использованием базовых алгоритмов, анализа типов коммуникаций и интеграции различных типов программного обеспечения

2. Показатели оценивания компетенций

В результате изучения дисциплины «Основы IT-технологий» обучающийся должен:

знать:

шаблоны проектирования программного обеспечения;
основы работы в UNIX-подобных системах;
основы низкоуровнего программирования;
основы машинного кода, языков ассемблера;
различные пути повышения производительности программы;
основы сетевого взаимодействия;
основы устройства сетей.

уметь:

работать с интерфейсом командной строки;
выполнять сборку программ из исходных текстов и их отладку, без использования интегрированных средств разработки;
пользоваться системами контроля версий;
настраивать окружение для непрерывной интеграции разработки проекта;
проектировать программное обеспечение таким образом, чтобы его поддержка осуществлялась коллективом из нескольких разработчиков;
создавать многопоточные и межсетевые программы на языках Си и Ассемблер;
работать в unix-подобных средах;
создавать программы на языках Си и Ассемблер без использования высокоуровневых библиотек.

владеть:

навыками работы с GitLab и GitLab CI
навыками ведения простейших программных проектов в системах контроля версий.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

Примеры контрольных заданий:

1. Написать программу, замеряющую время работы методов для нескольких стандартных реализаций интерфейсов List и Map, с использованием Dynamic Proxy.
2. Написать класс-наследник класса String, реализующий интерфейс Iterable, предоставляющий итератор со значениями типа char по своим символам.
3. Написать программу, которая скачивает 10 самых популярных страниц с сайта Хабрахабр и подсчитывает суммарный рейтинг комментариев, сагрегированный по пользователям, с использованием сторонней библиотеки парсинга HTML.
4. Написать программу, которая предоставляет консольный интерфейс для добавления записей в базу данных книжного магазина с сущностями «Автор» и «Книга» (отношение один-ко-многим между автором и книгами), с использованием сторонней библиотеки для работы с базой данных sqlite3.

5. Написать программу, которая предоставляет консольный интерфейс для добавления записей в базу данных книжного магазина с сущностями «Автор» и «Книга» (отношение один-ко-многим между автором и книгами), с использованием сторонней библиотеки, реализующей Object-Relation Mapping.

6. Написать программу, которая запрашивает у Foursquare API и выводит популярные кафе в окрестности города, вводимого пользователем.

7. Написать программу, вычисляющую PageRank и найти ТОП 100 самых популярных страниц в википедии (рускоязычной, англоязычной).

1. Реализуйте на языке ассемблера x86 (IA-32) или x86-64 функцию с сигнатурой:

extern double my_sin(double x)

которая вычисляет значение sin(x).

Запрещено использовать встроенные тригонометрические инструкции.

Для вычислений используйте известный вам из курса Математического анализа способ разложения функции в ряд. Точность результата должна быть маскимально возможной для типа данных double.

2. Программе в аргументе командной строки передается имя файла с бинарными данными в Little-Endian.

Файл хранит внутри себя односвязный список элементов:

```
struct Item {
  int value;
  uint32_t next_pointer;
};
```

Поле value хране значение элемента списка, поле next_pointer - позицию в файле (в байтах), указывающую на следующий элемент. Признаком последнего элемента является значение next_pointer, равное 0.

Расположение первого элемента списка (если он существует) - строго в нулевой позиции в файле, расположение остальных - случайным образом.

Выведите на экран значения элементов в списке в текстовом представлении.

Используйте отображение содержимого файла на память.

3.Программе задается единственный аргумент - номер TCP-порта.

Необходимо принимать входящие соединения на TCP/IPv4 для сервера localhost, читать данные от клиентов в текстовом виде, и отправлять их обратно в текстовом виде, заменяя все строчные буквы на заглавные. Все обрабатываемые символы - из кодировки ASCII.

Одновременных подключений может быть много. Использовать несколько потоков или процессов запрещено.

Сервер должен корректно завершать работу при получении сигнала SIGTERM.

Указание: используйте неблокирующий ввод-вывод.

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Перечень контрольных вопросов:

1. Сборщик мусора.
2. Массивы, многомерные массивы.
3. Методы класса Object. Контракт equals()/hashCode().
4. Интерфейсы.
5. Исключения: иерархия исключений, перехват исключений, декларация throws.
6. Порядок инициализации объекта.
7. Java Reflection API. Dynamic proxy.
8. Юнит-тестирование, средства JUnit.
9. Аннотации.
10. Итераторы.
11. Сериализация средствами стандартной библиотеки.

12. Лямбда-функции.
13. TCP/IP в Java. HTTP-сервер средствами Java.
14. Многопоточность: примитивы синхронизации в языке.
15. Многопоточность: Java Memory Model.
16. Многопоточность: инструменты библиотеки java.util.concurrent.
17. Коллекции стандартной библиотеки, потокобезопасность коллекций.
18. Форматы XML, JSON, CSV.
19. Кодировки текстовых данных.
20. Система контроля версий git.

Примеры билетов:

№1

1. Юнит-тестирование, средства JUnit.
2. Написать программу, замеряющую время работы методов для нескольких стандартных реализаций интерфейсов List и Map, с использованием Dynamic Proxy.

№2

1. TCP/IP в Java. HTTP-сервер средствами Java.
2. Написать класс-наследник класса String, реализующий интерфейс Iterable, предоставляющий итератор со значениями типа char по своим символам.

1. Система Linux, виртуальная машина. Инструменты для написания, компиляции и отладки программ.
2. Командный интерпретатор bash, написание shell-скриптов. Введение в язык Си.
3. Современный диалект языка Си (стандарт 2011 года). Отличия от C++, размещение данных в памяти, выравнивание данных, структуры и объединения, указатели на функции.
4. Представление целых чисел. Обратный дополнительный код, битовые операции. Знаковые и беззнаковые числа. Undefined Behaviour.
5. Язык ассемблера AVR, базовые конструкции; работа с регистрами и с памятью
6. Битовые операции на языке Си и ассемблере AVR; кодирование команд
7. Представление целых чисел, знаковые и беззнаковые числа, флаги переноса, длинная арифметика
8. Стек вызовов и прерывания
9. Язык ассебмлера ARM, базовые инструкции.
10. Стек вызовов и вызов функций на ARM
11. Представление вещественных чисел IEEE754
12. Программные прерывания и системные вызовы
13. Ассемблер x86_64. Архитектура CISC v.s. RISC. gdb и objdump. Соглашения о вызовах x86_64
14. Выравнивание данных и векторные инструкции SSE/AVX
15. Системные вызовы через int 0x80 и vdso (sysenter/syscall)
16. Файловые дескрипторы, open, read и write.
17. Системные вызовы POSIX для работы со временем: time, localtime, и пр. Проблема потокобезопасности.
18. Системные вызовы stat, access, readdir.
19. Отображение ELF файла на память; системный вызов mmap
20. Позиционно-независимый код и dlopen/dlsym
21. Системные вызовы fork, exec, exit
22. pipe, mkfifo, dup2 и межпроцессное взаимодействие
23. mmap и POSIX shm в качестве межпроцессного взаимодействия
24. Сигналы BSD и UNIX System V
25. Файловые дескрипторы signalfd и timerfd; механизм epoll
26. Posix Threads, мьютексы, семафоры и atomic

27. Условные переменные
28. Сокеты UNIX качестве межпроцессного взаимодействия
29. Сокеты TCP/IP. Сетевое взаимодействие
30. Прикладной уровень OSI. Протокол HTTP/1.1
31. Механизм epoll/kqueue для обработки TCP/IP
32. Сообщения UDP
33. Представительский уровень OSI. Шифрование с использованием Open/LibreSSL.

Примеры билетов (первая часть, третий семестр):

1. Стек вызовов и вызов функций на ARM. Файловые дескрипторы, open, read и write.
2. Стек вызовов и прерывания. Отображение ELF файла на память; системный вызов mmap.
3. Выравнивание данных и векторные инструкции SSE/AVX. Позиционно-независимый код и dlopen/dlsym.

Критерии оценивания

отлично

10 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы, код оформлен в едином удобочитаемом стиле

9 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы

8 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач

хорошо

7 Полностью решены все задачи. Допущены несущественные ошибки.

6 Полностью решено большинство задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

5 Полностью решено две трети задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

удовлетворительно

4 Полностью решено более половины задач. В остальных задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

3 Полностью решено более половины задач.

неудовлетворительно

2 Решено менее половины задач.

1 Не решено ни одной задачи.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Дифференцированный зачет может проводиться по итогам текущей успеваемости и сдачи заданий, лабораторных и других видов работ, предусмотренных программой дисциплины и (или) путем организации специального опроса, проводимого в устной и (или) письменной форме, а также с выдачей заданий для реализации на компьютере.