Разработка программных приложений

В написании использовался редактор `Atom`, а в качестве генератора `pdf` DILLINGER

Необходимо знать

Для освоение материала курса, студенту необходимо иметь начальные знания:

1. Архитектура ЭВМ;
2. Программирование на ЯПВУ Си;
3. Дискретная математика.

Цель курса

Целями данного курса являются:

1. Выбор и настройка окружения для разработки в зависимости от целевой ОС и задачи. Написание вспомогательных `batch` скриптов для упрощения взаимодействия с окружением и утилитами. Получение базовых навыков использования терминала `Unix-like` и `Windows` ОС. Работа с пакетными менеджерами.
2. Овладеть средствами автоматизации сборки, на примере утилиты `cmake`. Научиться по выданному заданию

распределить функциональные части приложения по модулям, определить вид модуля, границы его использования, конфигурирование, описание интерфейсов и т.д. Знать принципы непрерывной интеграции и доставки `CI/CD`.

3. Овладеть средствами контроля версий на примере `git`. Знать жизненный цикл кода в рамках использования контроля версий.
4.

Описание

1. История. Тьюринг, тезисы Черча. Вычислимость. Парадигмы программирования. История языка Си. C++ и Бьярн Страуструп с обобщенным программированием, как логическое продолжение языка. Стандарты языка. `POSIX`. Почему язык Си и почему стандарт `C90` (индекс TIOBE).
2. Окружение сборки. Окружение разработки. `IDE`, `JetBrains CLion`, `Microsoft Visual Studio Community Edition`. `MinGW-64`. Проект. Структура проекта. Автоматизация сборки. `cmake`. Непрерывная интеграция.
3. История систем контроля версий, `CVS`, `SVN`, `Mercurial`, `Git`. Базовые команды. `Index`, `Commit`, `Revert`, `Merge`, `Rebase`, `Pull`, `Push`, `Branch`, `Tag`. Последовательность работы. Разрешение конфликтов.
4. Парадигмы программирования: функциональное, процедурное, ООП, императивное, декларотивное, логической. Виды языков: статическая типизация, строгая типизация, динамическая типизация, компилируемость, транслируемость,

интерпритируемость (Примеры: `Asm`, `C`, `C++`, `Erlang`, `C#`, `Forth`). Виртуальная машина, `native` код, `JIT`. Процесс, стадии и порядок компиляции. Линковка, виды (статическая, динамическая). Динамически разделяемые библиотеки. Исполняемые модули. Виды: `ELF`, `PE`. Таблица импорта. Порядок загрузки исполняемого модуля(библиотеки). Базовый адрес.

5. Компилятор `gcc`, `cl`. Флаги компиляторов. Работа препроцессора. Организауия кода в языке Си: блок подключения заголовочных файлов, описание прототипов функций, описание глобальных переменных, описание реализации функций, точка входа. Типы данных. Структуры. Строки. Массивы. Переменный. Область видимость. Блок кода. Операции. Типы вызовов функций: `cdecl`, `stdcall`, `pascal`, `fastpascal`. Адрес возврата. Таблица функций (процесс компиляции). Описание конструкций языка: ветвления, циклы, переходы условные и безусловные.
6. Память. Типы памяти (статическая, динамическая, стек). Виды памяти в ОС: сегментная, страничная, сегментно-страничная. Виртуальная память. Помещение переменных в нужные сегменты. Получение доступа к сегментам. Средстав работы с динамической памятью в языке. `Embedded` как ограничение по ресурсам (в частности по памяти). Указатели. Представление памяти. Опасности при использовании прямого доступа к памяти. Опасняе функции. Виды атак. `Memory Leak`, отслеживание. `Valgrind`.
7. Базовые структуры: строки, массивы, списки, деревья, ассоциативные массивы. Базовые алгоритмы. Итераторы,

продолжения и т.д.

8. Файлы. Типы файлов в стандартной библиотеке и в ОС. Базовые функции работы с файлами. `Unix-like` - все является файлом. Дескриптор файла. Ограничение процесса в ОС. Стандартные потоки ввода, вывода. Демон, как специфичный объект в `Unix-like` ОС. Системный сервис в `Windows` - ответ `Unix-like` ОС. Виды доступа к файлу (последовательный, `random`). Перенаправление ввода вывода. Конвеер в терминале, базовые понятия (каналы и именнованые каналы далее). Атрибуты файлов, изменение атрибутов. `NTFS` потоки, чтение и управление.
9. Сокет. Типы сокетов (сетевые, `unix`). Сетевое взаимодействие. Модель `OSI`. Протколы `ARP`, `ICMP`, `TCP/IP`, `UDP`. Фрагментация и `MTU`. Порты. Сетевые адреса. Создание и базовые функции работы с сокетами (`Windows`, `Unix-like`). Блокирующие и не блокирующие сокеты. Конфигурирование сокетов `fcntl`.
10. Многопоточное программирование. Решаемые задачи, проблемы. Процессы, потоки, смена контекста, приоритеты. `pthread`, `fork`. Форк бомба. Создание, конфигурирование и управление потоком.
11. Синхронизация. Синхронизация потоков. Примитивы синхронизации: критические секции, спинлок, семафор, мьютексы, атомарные операции, `condition variable`, барьеры. Применимость. Проблемы: взаимная блокировка, использование `cpu` вместо ожидание, исчерпание ресурсов (дескрипторы потоков, стек потока). `Thread Local Storage`.
12. Межпроцессное взаимодействие. Предпосылки и задачи.

Примитивы: `FIFO`, `unix` сокеты, каналы (`fork` и наследование дескрипторов), `shared` память, блокировка файлов. Создание процесса и передача аргументов, специфика ОС. Мониторинг процесса. Управление процессом, сигналы, код возврата.

13. Тестирование. Модульное тестирование. Интеграционное. Эмуляция. `Mock`. Нагрузочное тестирование.

Порядок приема лабораторных работ

Порядок проведения практических работ