Идеи проектов

Данная страница является складом «сырых» идей проектов.

Марк Заславский

Эмулятор помещений для ROS / SLAM / CV задач

Цель - научится эмулировать данные от датчиков роботов (rgb и rgbd камеры, лидары), в помещениях различной пространтсвенной конфигурации (заданные 3d-моделями), с разными параметрами источников света (мощность, расположение, цветовая температура), расположением специальных меток (aruco, qr).

Задачи

- отображение 3d-моделей для помещений,
- конфигурация источников света,
- конфигурация расположения меток,
- конфигурация движения робота (в gui),
- конфигурация датчиков робота,
- экспорт измерений.

Результаты - программа с графическим интерфейсом, позволяющая формировать ros-bag файлы данных для заданных конфигураций, - технология быстрого подключения новых датчиков к программе (снятие параметров датчика).

[Done] Автоматизация проверки лабораторных для курса "Введение в нереляционные БД"

Цель: разработка системы автоматической проверки лабораторных работ для курса «Введение в нереляционные БД».

Задачи:

1. Изучение операций в нереляционных БД;
2. Разработка скриптов автоматизации для генерации условий и проверки лабораторных;
3. Разработка эталонных и ошибочных решений;
4. Интеграция наработок в stepic.org;

Требование:

1. Python, Linux

Тематики:

1. MongoDB
    1. полнотекстовый поиск
    2. геозапросы
    3. GridFS
    4. MapReduce
2. ArangoDb
    1. Создание объектов БД всех видов
    2. Графовые операции
3. Cassandra
    1. установка и настройка
    2. создание БД
    3. запросы на чтение/запись
    4. BigTable специфика
4. OpenLink Virtuozo
    1. SPARQL запросы
    2. RDF
5. ExistDB

Результат: набор автоматически проверяемых заданий.

http://se.moevm.info/doku.php/staff:courses:no_sql_introduction

Генераторы задач

[WiP] Программирование

Прямые и обратные задачи на:

1. порядок отладки,
2. сообщения компилятора,
3. синтаксические ошибки

[WiP] Курс по инструментам отладки и профилирования (gdb, vallgrind, callgrind)

Интерактивные задачи на дебаг:

1. сегфолтов
2. неправильных вызовов
3. ошибок преобразования типов
4. поиск ошибок параллельного исполнения
5. обратная инженерия
6. проблемы с производительностью

Научиться «замусоривать» код программы кодом без побочных эффектов.

Задачи GDB:

- Сегфолты
  - Дан бинарный файл программы, собранный с отладочными символами. Программа падает при запуске с сегфолтом. Необходимо найти имя переменной и функции, где это происходит.
  - Тоже что и выше, только программа запущена в режиме «демона» (висит в памяти)(проверка на владение gdb attach).
- Исследование работоспособных программ:
  - Определение места в коде, где допущена ошибка.
    - Дан бинарный файл программы, собранный с отладочными символами. Исходный код выглядит достаточно непонятно (обфускатор). Программа реализует (варианты):
      - вычисление суммы чисел от 1 до N (либо какой-то похожий простой алгоритм) и делает это неправильно.
      - запись данных в файл, но почему-то данные в файл попадают не все.
      - сортировка строк, которая работает не правильно.
    - Задача: с помощью gdb установить, где ошибка (номер строки/имя функции).
  - Поиск значения в памяти:
    - Дан бинарный файл программы, собранный с отладочными символами. Исходный код выглядит достаточно непонятно (обфускатор). Программа реализует итеративное вычисление какого-нибудь значения (строкового или числового):
      - контрольная сумма
      - шифрование
      - приближенное значение мат. функции
    - Задача: с помощью gdb установить, значение функции на итерации N.

Задачи valgrind:

- Утчека памяти. Дан бинарный файл программы, в котором допущена утечка памяти. Задача: определить переменную, через которую утекает память.
- Производительность. Дан бинарный файл программы, в котором ооооочень много разных функций и их вызовов. Задача: определить самую часто вызываемую функцию.

[WiP] Система автоматической проверки наиболее частых формальных ошибок в научных статьях/отчетах

Вводная часть: Разработать веб-сервис, который проводит анализ текста научной статьи/студенческого отчета с помощью систем полнотекстового поиска для проверки критериев. Критерии включают самые типичные (но при этом машинно-проверяемые) ошибки при подготовке данных документов, например:

1. личные предложения и личные формы глаголов
2. отсутствие ссылок или битые ссылки на элементы списка литературы/рисунки/таблицы
3. повторы слов в пределах двух предложений
4. «телеграфность»
    1. повторение начальных слов абзацев («Было принято решение»)
5. стоп-слова:
    1. жаргонизмы (использовать список): скачать, пост, либа, тул
    2. личные местоимения

3. там, тут, здесь

Задачи:

1. парсинг docx/ppt/pdf
2. полнотекстовый поиск
3. проверка выполнения больших наборов стандартизированных правил для текста на естественном языке

[WiP] Kernel programming

Цель: разработка проверяющих скриптов для заданий курса «Программирование в ядре Linux».

Задачи:

1. изучение механизмов работы ядра Linux и способов его модификации
2. создание и отлаживание проверяющих скриптов для задач по разработке модулей ядра/драйверов/ прочих расширений)
3. создание эталонных и ошибочных решений для задач

Требования:

1. знание программирования ядра на начальном уровне
2. C, Make, Linux

Результат: набор сценариев проверки, заинтегрированных в онлайн версию курса.

[Done] Сбор статистики курса "Основы программирования в Linux"

Цель: доработка и расширение функционала статистического фронтенда

Задачи:

1. изучение принципов статистического анализа в связке Python+Mongo
2. полнотекстовый поиск силами mongo
3. сбор и вычисление статистики курса (самые проблемные задачи/как быстро и часто решают и тд)
4. динамическая подгрузка лога
5. адаптивная и отзывчивая графика
6. генерация/экспорт отчетов

Требования:

1. Python, Django
2. JS библиотеки для визуализации графики

Результат: веб-сервис, который позволяет вести мониторинг статистики прохождения курса и отслеживать определенные события в логе.

[Done] Инструмент мониторинга статистики прохождения outdoor-квестов

Цель: создать веб-приложения для исследования подробной статистики прохождения квестов + реализовать интерфейсы сбора подобной статистики для приложений.

Задачи:

- Исследовать примерные аналоги
- Спроектировать веб-интерфейс, в котором можно будет просматривать и визуализировать подробную статистику.

Требования:

- Python + turbogears
- JS-фреймворки для визуализации статистики
- mongodb

Результат: веб-сервис, который позволяет владельцу набора квестов просматривать статистику их прохождения.

Доработка ИС кафедры для учета дипломников

Цель: создание дополнительной подсистемы к ИС кафедры для учета дипломников

Задачи:

1. реализовать набор view для CRUD дипломной информации (руководитель, тема, ссылки на материалы, публикации, документы (ПЗ, отзыв, задание, рецензия)) с учетом сроков ее предоставления
2. реализовать сводное view прогресса работы над дипломом (по аналогии с таблицей дипломников)
3. реализовать набор view для редактирования критериев аттестации дипломников (сроки)
4. Интерфейс для рецензирования дипломов

Требования:

1. максимально использовать существующие наработки (в особенности БД)
2. python / django

[Done] Тренажер публичных выступлений

Цель: сделать тренажер, позволяющий докладчику объективно измерить

1. скорость речи (сколько слов в секунду он произносит)
2. четкость речи
3. скорость доклада (расход времени/слов на каждый слайд/ среднее время/слова на каждый слайд)
4. проверить укладываемость во временное ограничение

5. измерить теже самые параметры в контексте ответов на вопросы
6. сопоставить распознанные слова и текст речи или тезисы для выявления неосвященных тем*

Предполагаемая форма исполнения - веб-приложение.

Сценарий использования:

1. пользователь открывает приложение
2. пользователь загружает презентацию и указывает временное ограничение (количество минут на доклад)
3. пользователь нажимает кнопку «Начать тренировку»
4. на экране отображается презентация, обратный отсчет времени, номер слайда/общее количество слайдов, график (стрелочный индикатор) количества слов в минуту , кнопки переключения слайдов
5. пользователь осуществляет доклад
6. если темп речи превышает некоторый, заранее заданный порог, то график/индикатор окрашивается красным
7. по окончанию доклада пользователь нажимает кнопку «Доклад окончен»
8. на экране отображается статистика выступления - общая и по отдельным слайдам

[Done] Виртуальный тренажер Arduino

Цель: реализовать набор автоматически проверяемых (не интерктивных) задач для обучения программированию Arduino.

Задачи:

- создание Docker-образа для эмулятора Arduino,
- автоматизация запуска эмулятора Arduino,
- сборка студенческих решений и проверка корректности их работы по сценарию задачи.

Требования:

- начальные знания разработки для Arduino,
- минимальный опыт работы с Docker либо Vagrant,
- базовый навык работы с Bash.

Результат: набор задач, опубликованных в курсе на Stepik.org.

Построитель схем для SimulAVR

Цель - реализовать консольный редактор схем для SimulAVR с поддержкой простых компонентов:

- диоды,
- резисторы,
- лампочки,
- конденсаторы,

- катушки,
- фоторезисторы и другие.

Редактор должен позволять:

- редактировать, сохранять в файл и визуализировать схемы;
- настраивать параметры элементов, в том числе задавать сигналы вручную;
- запускать программы в рамках этих схем;
- визуализировать сигналы.

Требования:

- Python
- Bash
- SimulAVR

Результат: консольное приложение.

[WiP] Эмулятор arduino для ArduinoIDE

Тоже что и в предыдущей теме, но GUI + интеграция к IDE.

Мини-курсы Stepik

- по созданию и публикации Gem
- по созданию и публикации Vagrant плагинов
- как создавать Docker образы
- как пользоваться screen
- как искать файлы и делать grep
- как пользоваться man

Права доступа в linux

- рассказать о том, как
  - устроены права, пользователи и группы
  - что же делает судо и как его настроить
  - исполняемые файлы и права на запуск
  - как решать типичные ошибки (нет прав/ файл не найден/отказано в доступе)
- примеры задач:
  - соотнесите цифровые и буквенные права
  - настройте права так, чтобы определенные пользователи/группы имели/не имели доступа

[Done] ssh-tricks

- Интерактивные задачи (генератор среды + скрипты проверки):

- выполняем команды на удаленном сервере
- генерируем ключи
- настраиваем авторизацию без пароля (authorized keys)
- настраиваем авторизацию по ключу (-i)
- учимся делать туннели
- scp
- sshfs

Мат.модели работы ПО

у студентов часто возникает задача в дипломе / курсово исследовать типовые показатели работы программы: - расход памяти - скорость работы - пропускная способность (скорость передачи данных по сети)

При этом студенты демонстрируют очень низкий уровень понимания того как ставить эксперимент, обрабатывать данные и тд. Есть гипотеза что даже хорошее преподавание статистики/метрологии не поможет сильно исправить ситуацию, так как там материал зачастую оторван от жизни.

Хочется сделать курс, где на предельно прикладных задачах (буквально надерганных с дипломов/курсовых) будет показано как применять мат. модели, но без слишком большого объема теории (а лучше - с минимумом). Например: - как построить зависимость скорости работы программы (дается студенту в виде бинарника/исходника) от параметра A – какие там будут источники погрешностей, как их исключить/оценить/проверить гипотезы про распределения – какой метод интерполяции выбрать? – как поставить эксперимент по измерению? – как обрабатывать полученные данные? – какие выводы можно, а какие нельзя делать по результатам?

Курс по развитию внимательности

Как известно, 90% проблем у учащихся возникает при невнимательном чтении заданий. Необходимо сделать такой курс, чтобы окончив его, человек умел читать тексты максимально внимательно + анализировать их содержимое.

[Done] Автоматизация проверки заданий для XV6 по курсу ОС

Цель: реализовать автоматически проверяемые задачи по системе XV6 на базе проверяющей системы «Основы программирования в Linux».

Задачи:

1. составление эталонных решений к задачам;
2. создание скриптов проверки.

Требования:

1. базовое понимание того, как работают ОС (более высокий уровень знания или понимания будет плюсом);
2. владение C;
3. базовые знания Ruby, Bash;
4. общее предсталение о работе Vagrant.

Результат: набор автоматически проверяемых задач, заинтегрированных на Stepik.org.

Ссылки:

1. https://pdos.csail.mit.edu/6.828/2012/xv6.html
2. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-828-operating-system-engineering-fall-2012/index.htm

Плагины для jenkins

- Массовая настройка слейвов (одновременный запуск на N слейвах, анализ кодов возврата и вывода).
- Запуск сложных конфигураций BM для job.

[Done] Плагины vagrant

- Диагностика запуска BM.
- Поддержка ограниченного пула портов|директорий.
- Создание отладочных копий BM в случае сбоя.
- Быстрая настройка доступа по ssh-key.

Поиск плагиата по комментариям в исходном коде

Дано - исходный код в git репозитории. Задача - найти другие репозитории, в которых есть такие же комментарии в исходных кодах.

Расширения для хрома, упрощающие работу со Stepik

- Фильтрация по табелю успеваемости.
- Отображение процента прохождения курса участником в комментарии:
  - сколько прошел всего,
  - сколько было попыток решить данный степ,
  - ссылка на последнее решение.

[Done] Автоматизация ответов на вопросы в рамках

онлайн-программы

Цель: создать инструмент, ускоряющий ответы на вопросы участников онлайн-программы.

Задачи:

1. экспорт данных из задач Redmine
2. индексация
3. выделение ключевых слов
4. построение рекомендаций

[Done] Поиск/автоматическое удаление словестных пауз из видео

Очень часто при записи онлайн-курсов авторы видео делают много пауз для того, чтобы собраться с мыслями/обдумать следующее слово. Задача - найти и удалить их.

Расширение для Google Chrome, подсчитывающее статистику изменений задач в Trello-доске

Интеграция задач на Scratch в Stepik

- Портировать веб-фронтенд Scratch в удобную/контролируемую форму.
- Создать (скорректировать) интерфейсы для проектирования задач.
- Связать фронтенд со Stepik, так, чтобы решение задач во фронтенде приводило к соответствующим отметкам на Stepik.

Задачи для курса по ядру / программированию в Linux

Онлайн-курс для аспирантов

Идея - дать представление об общих основных этапах подготовки к защите и улучшить понимание следующих разделов:

- соответствие паспорту специальности,
- научные результаты,
- в целом что можно и нельзя писать.

Автоматизация заполнения шаблонов документов

Цель - инструмент, который позволит автоматически заполнять шаблоны документов (docx, odt, pptx, odp ….) данными из таблицы (каждая колонка - отдельное подставляемое поле, каждая строчка - новый документ).

Выходные данные:

- CSV-файл со значениями подстановок,
- путь к шаблону в формате docx, odt, pptx, odp,
- текстовый шаблон именования файла.

Пример CSV-файла:

```
FirstName,LastName,Adress,City,State,ZIP-code
John,Doe,120 jefferson st.,Riverside, NJ, 08075
Jack,McGinnis,220 hobo Av.,Phila, PA,09119
"John ""Da Man""",Repici,120 Jefferson St.,Riverside, NJ,08075
Stephen,Tyler,"7452 Terrace ""At the Plaza"" road",SomeTown,SD, 91234
,Blankman,,SomeTown, SD, 00298
"Joan ""the bone"", Anne",Jet,"9th, at Terrace plc",Desert City,CO,00123
```

Соответственно, полями для подстановки являются: FirstName, LastName, Adress, City, State, ZIP-code.

В файле docx, odt, pptx, odp содержатся указанные выше поля подстановки в виде меток следующего вида:

```
{{ Adress }}
```

Текстовый шаблон именования файла содержит комбинацию полей подстановки:

```
{{ FirstName }}{{ LastName }}.docx
```

В результате работы должны появится 6 файлов, по одному для каждой строчки CSV файла.

Фреймворки:

- https://docxtpl.readthedocs.io/en/latest/
- https://pypi.org/project/python-pptx-templater/
- https://pypi.org/project/pptx-template/

Кирилл Кринкин

- Генерация отчетных форм по лабораторным и курсовикам
- Автоматическая проверка отчетов присылаемых на заданный емайл и сортировка их по папкам (по мотивам http://se.moevm.info/doku.php/start:report_submission)

Интеграция задач на Scratch в Stepik

- Портировать веб-фронтенд Scratch в удобную/контролируемую форму.
- Создать (скорректировать) интерфейсы для проектирования задач.
- Связать фронтенд со Stepik, так, чтобы решение задач во фронтенде приводило к соответствующим отметкам на Stepik.

Татьяна Берленко

Stepic

Комментарии

Задачей является создание новых или доработка существующих инструментов, которые бы позволили вести учет студенческих комментариев на различных курсах Stepic. Необходимо отслеживать ветки дискуссий и сохранять в БД информацию о не обработанных комментариях.

Статистика

Скрипт, который на основе .csv файла с результатами прохождения модуля/курса студентов строит статистику.

Предоставление фидбэка студентам

На основе решения студента, функции-генератора test case, функции решения и проверки получаем полный фидбэк (место ошибки)

Получение информации о коде студента

Проверка кода студента на выполнение неких правил задачи:

1. наличие определенных функций
2. невмешательство в изначальный код, который был дан преподавателем
3. использование каких-то определенных методов/инструментов языка

Проверка на жульничество

Сводная таблица о тех кто жульничает.

MSE

Глобальная идея: собрать все наработки по mse в одно место.

Предоставление информации об окончании этапа кураторами

Веб-форма, в которой в конце этапа куратор указывает следующее:

1. что необходимо было сделать
2. что было сделано
3. оценку каждому участнику

После отправки формы кураторы и организаторы получают письмо, в котором вложена google table с оценками и сводная таблица с информацией (что было сделано, что должно было быть сделано) для простоты оценивания

Geo2Tag

Map

- Закончить реализацию отсюда Вики с описанием интерфейса доступа к /map
- Реализация для карты параметра «hightlight_point», с помощью которого на карте будет выделяться другим цветом точка, id которой передаётся в этом параметре.

Логгирование

- Добавить в логи тип запроса (CRUD)

Unassigned

Веб-интерфейс для инструмента версионируемой загрузки курсов на Stepik

Цель: создание веб-интерфейса к инструменту командной строки https://github.com/OSLL/stepic_uploader , позволяющего автоматизировать создание уроков и курсов в рамках платформы Stepik. Реализуемые сценарии использования:

1. Создание урока-контрольной по существующему набору вопросов с выбором ответа.
2. Загрузка и выгрузка курса в машиночитаемом виде в систему контроля версий.

Задачи:

1. Создание пользовательского интерфейса.
2. Подключение авторизации с использованием данных Github и Stepik по протоколу OAuth2.
3. Автоматизация установки сервиса и настройки программной среды.

Требования:

1. Python, Flask, JS

Результат: веб-сервис загрузки и выгрузки для курсов на stepik.org .

Создание задач по письмам meeting minutes

TBD