Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки «Программная инженерия»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

курса «Информатика»

Вариант 3

Выполнил студент:

Берестовский Святослав Сергеевич
группа: P3111

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург, 2021 г.

Содержание

Лабораторная работа 4

Исследование протоколов, форматов обмена информацией и языков разметки документов

4.1. Предисловие

В ходе выполнения данной лабораторной работы, мне, так или иначе, приходилось осуществлять валидацию примитивов разных языков разметки. В то время как `JSON` или `XML` принято обрабатывать в соответствии со стандартизацией, `HTML`, в силу особенностей применения, позволяет допускать некоторые неточности при верстке. Дело в том, что алгоритмы синтаксического анализа, заложенные в современных браузерах, имеют механизмы обработки частых ошибок.

По очевидным причинам я посчитал неразумным реализовывание данных механизмов в рамках поставленных задач.

4.2. Описание заданий

4.2.1. Обязательное задание

1. Понять устройство страницы с расписанием для своей группы: https://itmo.ru/ru/schedule/0/P3111/schedule.htm
2. Исходя из структуры расписания конкретного дня, сформировать файл с расписанием в формате, указанном в задании в качестве исходного.
3. Написать программу на языке `Python 3.x`, которая бы осуществляла парсинг и конвертацию исходного файла в новый без использования готовых библиотек, в том числе регулярных выражений в `Python` и библиотеки дйля загрузки XML-файлов.

4.2.2. Дополнительное задание задание №1

1. Найти готовые библиотеки, осуществляющие аналогичный парсинг и конвертацию файлов.
2. Переписать исходный код, применив найденные библиотеки. Регулярные выражения также нельзя использовать.
3. Сравнить полученные результаты и объяснить их сходство/различие.

4.2.3. Дополнительное задание задание №2

1. Переписать исходный код, добавив в него использование регулярных выражений.
2. Сравнить полученные результаты и объяснить их сходство/различие.

4.2.4. Дополнительное задание задание №3

1. Используя свою исходную программу из обязательного задания, программу из дополнительного задания №1 и программу из дополнительного задания №2, сравнить десятикратное время выполнения парсинга + конвертации в цикле.
2. Проанализировать полученные результаты и объяснить их сходство/различие.

4.2.5. Дополнительное задание задание №4

1. Переписать исходную, чтобы она осуществляла парсинг и конвертацию исходного файла в любой другой формат (кроме `JSON`, `YAML`, `XML`, `HTML`): `PROTOBUF`, `TSV`, `CSV`, `WML` и т.п
2. Проанализировать полученные результаты, объяснить особенности использованного формата.

4.3. Обязательное задание

4.3.1. Комментарий

Реализацию поставленной задачи я начал с выделения необходимых компонентов:

- Инструментарий для парсинга `HTML`
- Инструментарий для парсинга `JSON`
- Инструментарий для сборки `YAML`

Парсинг HTML

Имея опыт взаимодействия с библиотеками для парсинга `HTML`, я решил ограничиться двумя функциями: `extract` и `remove`. Первая осуществляет поиск и извлечение содержимого всех тегов с указанным именем, вторая удаляет теги, оставляя их содержание. Фактически же функция `extract` посимвольно считывает исходный текст, находя нужные теги и сохраняя их позиции в тексте, объединяет открывающие и закрывающие теги в пары с учетом вложенности и генерирует список их содержимого. Функция `remove` также посимволно считывает исходный текст, однако встречая теги, просто инорирует их.

Парсинг JSON

Для парсинга `JSON` я придумал рекурсивный алгоритм: сначала просходит посимвольное считывание исходного текста с разбинием его на отдельные элементы. Далее определяется тип элемента и, если он является объектом `JSON` или массивом, передается в соотвествующую функцию. Таким образом максимальная глубина `JSON` ограничена лишь максимальной глубиной рекурсии.

Сборка YAML

Алгоритм сборки `YAML` обратен алгоритму парсинга `JSON`: осуществяется перебор элементов словаря, определяется их тип. В случае если элемент является вложенным, он передается в соотвествующую рекурсивную функцию, в противном случае происходит форматирование элемента в соответствии с синтаксисом `YAML`.

4.3.2. Исходный код

Исходный код всех скриптов доступен на моем GitHub: https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-4/task-1/.

4.4. Дополнительное задание задание №1

4.4.1. Исходный код

Исходный код всех скриптов доступен на моем GitHub: https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-4/task-2/.

4.4.2. Использованные библиотеки

- `Beautiful Soup`: https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/bs4ru.html
- `PyYAML`: https://pyyaml.org/wiki/PyYAMLDocumentation
- `json`: https://docs.python.org/3/library/json.html

4.5. Дополнительное задание задание №2

4.5.1. Исходный код

Исходный код всех скриптов доступен на моем GitHub: https://github.com/anim3boy/itmo-labs/blob/main/computer-science/lab-4/task-3/.

4.6. Дополнительное задание задание №3

4.6.1. Расчет времени выполнения

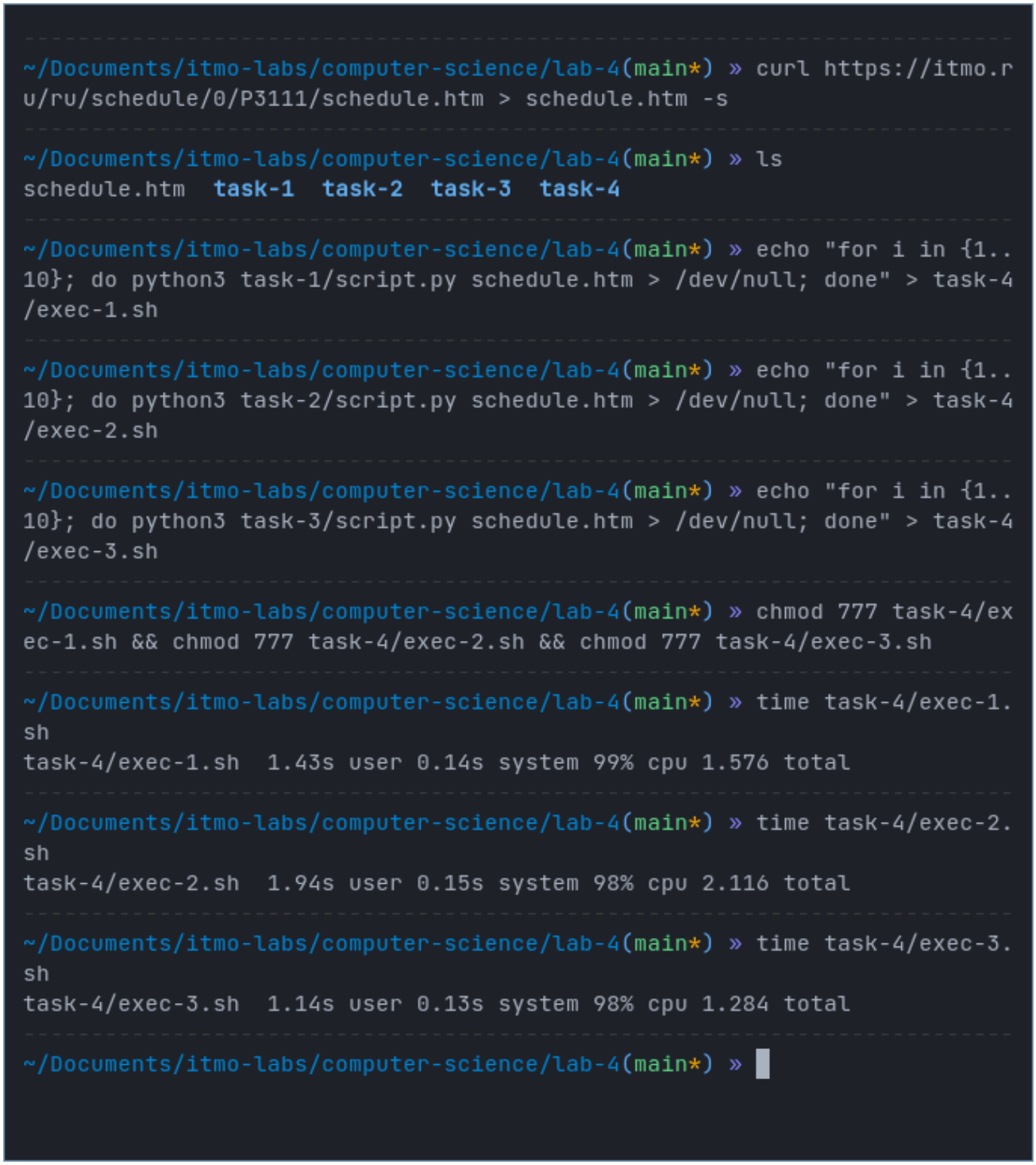

Рис. : Скриншот фрагмента сессии в ZSH

4.6.2. Краткий анализ

Скрипт использующий регулярные выражение работает быстрее остальных. Следующая по времени реализация использует сторонние библиотеки. Могу

предположить, что это связано с коррекцией ошибок при парсинге `HTML`. Медленнее всего работает алгоритм, считывающий текст посимвольно. Это ожидаемо, т.к он использует ряд неоптимальных решений.

4.7. Вывод

В ходе выполнения данной лабораторной работы я детально разобрался с устройством популярных языков разметки, потренировался в написании регулярных выражений, придумал несколько алгоритмов, в том числе с использованием рекурсии. Особенно порадовал серьезный и комплексный подход к разработке задания к данной лабораторной работе - страница с расписанием группы намеренно презентует студентам пример плохой верстки и способность браузера угадывать положение пропущенных закрывающих тегов.

Литература

[1] Regular Expression HOWTO [Электронный ресурс] // Python Documentation — https://docs.python.org/3/howto/regex.html — (дата обращения: 10.10.2021).

[2] Регулярные выражения в Python от простого к сложному. Подробности, примеры, картинки, упражнения. [Электронный ресурс] // Хабр — https://habr.com/ru/post/349860/ — (дата обращения: 10.10.2021).