Анализ разработанных задач в системе автоматической проверки для МООС «Программирование модулей ядра Linux»

Канушин Максим

21 декабря 2017 г.

1 Аннотация

В данной работе исследуется статистика прохождения студентами массового открытого онлайн курса «Программирование модулей ядра Linux»[1]. Проверка заданий, присланных студентами проходит в автоматическом режиме разработанной ранее системой. Задания представляют собой лабораторные работы, в рамках которых студенту необходимо разработать программу на языке C[2] и Makefile[3]. Целью исследования является определить насколько хорошо разработанные задачи и система их проверки соответствуют видео-лекциям и какие действия стоит предпринять, чтобы уменьшить количество студентов, бросающих онлайн-курс.

2 Введение

Многие студенты бросают онлайн-курсы когда перестают справляться с предложенными им заданиями или из-за того, что курс им надоедает и практические задания перестают быть им интересны. Для онлайн-курса «Программирование модулей ядра Linux» была разработана система, проверяющая корректность написанных студентами модулей ядра, которая позволила участникам курса загружать свои решения и в течение нескольких десятков секунд получать результаты проверки. Для того, чтобы улучшить онлайн-курс и уменьшить отток студентов, была проанализирована статистка тестового запуска курса. Целью исследования статистики являлось определить какие задачи тяжело даются студентам, а какие - наоборот легко и какие факторы могут на это влиять.

3 Описание проверяющей системы

Система автоматической проверки лабораторных работ представляет из себя комплекс взаимосвязанных программных средств, позволяющих студенту загрузить разработанную им программу и получить ответ о ее корректности, а так же логи сборки, запуска и информацию о произошедших ошибках при их наличии. Система обеспечивает изолированность проверяемого решения от внешней среды, чтобы обеспечить достаточный уровень надежности и безопасности исполнения. Помимо этого исключается взаимное влияние решений друг на друга для того, чтобы не дать возможность студенту случайно или намеренно повлиять на работу другого студента или просмотреть его. Среда, в которой происходит проверка, одинакова для всех решений, чтобы обеспечить как одинаковые условия проверки различных решений, так и многократную воспроизводимость результатов проверки одного и того же решения.

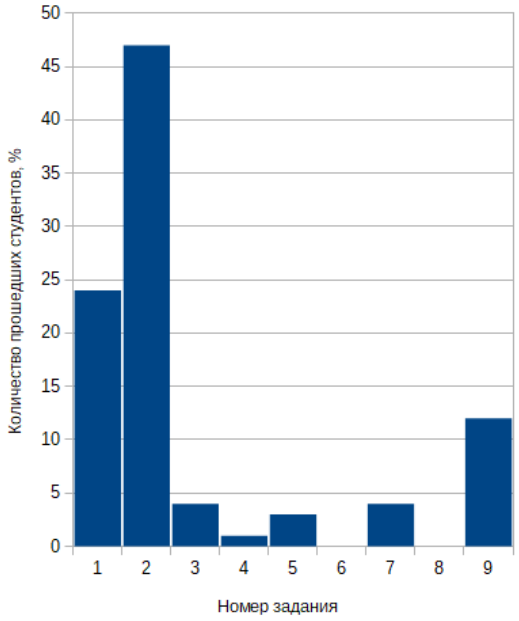
4 Обзор результатов прохождения онлайн-курса

Онлайн курс «Программирование модулей ядра Linux» был запущен в тестовом режиме. Курс состоял из 9 практических задач. В сумме по всем задачам было получено 795 решений из которых 270 правильных.

Статистика прохождения онлайн-курса представляла из себя csv-файл, содержащий информацию о всех загруженных решениях. Для ее обработки был создан набор скриптов на языке программирования Python[4]. Для изучения статистики использовались следующие данные: идентификатор пользователя, идентификатор решаемой задачи, результат проверки и логи сборки, скрипта проверки и запуска решения. Скрипт позволил получить необходимые поля из csv-файла и посчитать метрики, описанные ниже.

4.1 Stopout

Рис. 1: Stopout



Stopout[6][5] - величина, показывающая, сколько человек прекратили прохождение курса на той или иной задаче. Исследование этой характеристики позволит понять, какие задачи стоит усложнить, а какие, наоборот - сделать проще.

На рис. 1 представлен график, показывающий распределение студентов, бросивших курс, по задачам до которых они добрались.

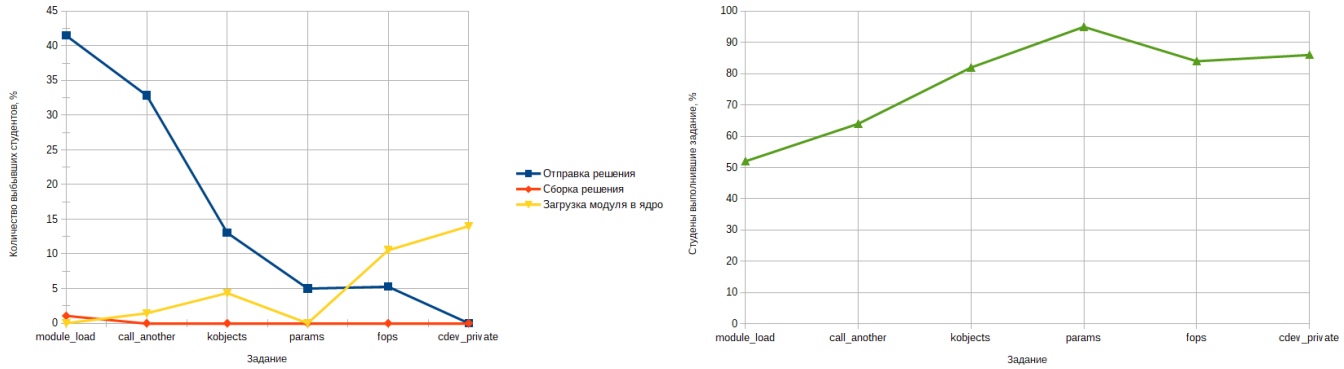
Первые два задания являются довольно простыми и схожими по сложности, что объясняет, почему многие студенты, выполнившие первое задание, решили и второе. Третье задание было заметно сложнее и большинство студентов бросили курс именно на ней. Данная метрика позволила понять, что разница в сложности между второй и третьей задачей слишком велика, что оттолкнуло студентов от продолжения курса.

4.2 Пройденные студентами этапы в рамках задач

По результатам проверки решений, были подсчитаны этапы в рамках каждой задачи, которые удалось пройти студенту. В общем случае выполнение любой задачи по курсу можно было разбить на 4 этапа: отправку файла с исходным кодом, сборку, загрузку собранного модуля в ядро и выполнение поставленных условий задачи. В ходе разработки скриптов и анализа статистики было обнаружено, что система в своем текущем виде не позволяет разбить задачу на большее число этапов, что позволило бы подробно исследовать проблемы каждой из задач.

Для каждого студента был посчитан шаг, на котором он бросил ту или иную задачу. На рис. 2a представлен график, показывающий какой процент студентов остановился на том или ином этапе задачи. Прохождение последнего этапа означает, что написанный модуль ядра полностью рабочий и выполняет все требования задания. Распределение по этому этапу выделено в отдельный график, представленный на рис. 2b. Данные, представленные для каждой задачи на обоих графиках, высчитывались относительно только тех студентов, которые приступили к ее выполнению.

Из графика на рис. 2b видно, что с каждой новой задачей, все больше студентов, которые брались за ее выполнение, старались довести ее до конца. Так же на графике 2a можно заметить, что с каждой новой задачей у студентов стали лучше получаться первые шаги, то есть все меньше студентов бросали решение задачи на неудавшейся сборке решения или загрузке модуля в ядро, и многие застревали именно на последнем, самом сложном этапе, что означает, что разработанная система позволила многим студентам отработать общий навык написания модулей ядра и уменьшить количество совершаемых ими ошибок.



(a) (b)

Рис. 2: Распределение студентов по этапам, на которых они бросили задачу

Поэтапное исследование решений задачи позволило понять, что в первых двух задачах курса, большинство студентов бросили задачу на самом первом этапе - неудавшейся сборке решения. Значительная разница в сложности между второй и третьей задачей курса оказалась не связана ни с одним из этапов их решения, так как большая часть студентов, взявшихся решать третью задачу, довели ее до конца. Данное наблюдение позволяет понять, что упрощение третьей задачи не даст должного эффекта и отток студентов не уменьшится. Вместо этого является целесообразным добавить дополнительную задачу, которая позволила бы студентам лучше усвоить тему и совершить более плавный переход к последующей задаче.

5 Заключение

В данной работе была изучена статистика тестового запуска онлайн-курса «Программирование модулей ядра Linux». Анализ этой статистики показал, что большая часть студентов бросала курс дойдя до той или иной задачи, но не приступая к ее решению. Исследование этапов, на которых застряли студенты, позволило понять, что для улучшения задач курса необходимо доработать систему таким образом, чтобы для каждой задачи можно было выделить большее число этапов ее решения, чтобы понять где именно у студентов возникает больше всего проблем. Было решено, что после второй задачи целесообразно добавить еще одну - промежуточную, с целью уменьшения боязни студентов условий последующих задач и их оттока с курса.

Список литературы

[1] Разработка модулей ядра Linux, // Stepik, URL: https://stepik.org/course/2051 [Посещено: 15 Дек 2017]

[2] Kernighan, Brian W.; Ritchie, Dennis M., The C Programming Language (1st ed.)., Englewood Cliffs 1978

[3] GNU Make, URL: https://www.gnu.org/software/make/ [Посещено 17 Окт 2017]

[4] Python, URL: https://www.python.org/ [Посещено: 8 Ноя 2017]

[5] Taylor, C. and Veeramachaneni, K. and O'Reilly, U.-M. *Likely to stop? Predicting Stopout in Massive Open Online Courses*. ArXiv e-prints, 2014.

[6] Merriam-Webster dictionary, URL: https://www.merriam-webster.com/dictionary/ [Посещено: 28 Ноя 2017]