Анализ Разработанных Задач в Системе Автоматической Проверки для МООС «Программирование Модулей Ядра Linux»

Канушин Максим
Санкт-Петербургский Государственный
Электротехнический Университет
СПбГЭТУ «ЛЭТИ»
Email: rextuz@gmail.com

Аннотация—В данной работе исследуется статистика прохождения студентами массового открытого онлайн-курса «Программирование модулей ядра Linux». Проверка заданий, присланных студентами проходит в автоматическом режиме разработанной ранее системой. Целью исследования является определить, насколько корректен выбор сложности разработанных задач, а так же понять, как стоит модифицировать задачи, чтобы уменьшить отток студентов с онлайн-курса.

I. Введение

Для онлайн-курса «Программирование модулей ядра Linux» [1] была разработана система, проверяющая корректность написанных студентами модулей ядра, которая позволила участникам курса загружать свои решения и в течение нескольких минут получать результаты проверки. Практические задания курса представляют собой лабораторные работы, в рамках которых студенту необходимо разработать программу на языке C [2] и Makefile [3]. 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.

II. Описание проверяющей системы

Система автоматической проверки лабораторных работ [4] 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.

III. Обзор результатов прохождения онлайн-курса

Онлайн-курс «Программирование модулей ядра Linux» был запущен впервые в тестовом режиме в июле 2017г на платформе Stepik [5], после чего курс открывался еще несколько раз.

Список практических задач курса представлен в табл. 1. В состав курса вошло 9 практических задач. В сумме по всем задачам было получено 795 решений из которых 270 правильных.

Статистика прохождения онлайн-курса была получена средствами платформы Stepik и представляла из себя csv-файл, содержащий информацию о всех загруженных решениях, которая для каждого решения включала идентификатор пользователя, идентификатор решаемой задачи, результат проверки, логи сборки, скрипта проверки, запуска решения и др. Для ее обработки был создан набор скриптов на языке программирования Python [6]. Скрипт позволил посчитать метрики, описанные ниже, которые позволили проанализировать прогресс студентов по курсу и в рамках одной задачи, что было необходимо для оценки разработанных задач.

A. Stopout

Stopout [8] [7] - величина, показывающая, сколько человек прекратили прохождение курса на той или иной задаче. Исследование этой характеристики позволило понять, какие задачи стоит усложнить, а какие, наоборот - сделать проще.

На рис. 2 представлен график, показывающий распределение студентов 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 прекратили прохождение курса именно на ней. Данная метрика позволила понять, что

Порядковый номер	Идентификатор	Ссылка
1	module_load	https://stepik.org/lesson/49216/step/1
2	call_another	https://stepik.org/lesson/49216/step/2
3	kobjects	https://stepik.org/lesson/40444/step/2
4	params	https://stepik.org/lesson/40444/step/3
5	dynamic_node	https://stepik.org/lesson/40444/step/4
6	fops	https://stepik.org/lesson/40451/step/1
7	cdev_private	https://stepik.org/lesson/40451/step/2
8	linked_lists	https://stepik.org/lesson/40456/step/1
9	ioctl	https://stepik.org/lesson/40456/step/2

Рис. 1. Практические задачи курса

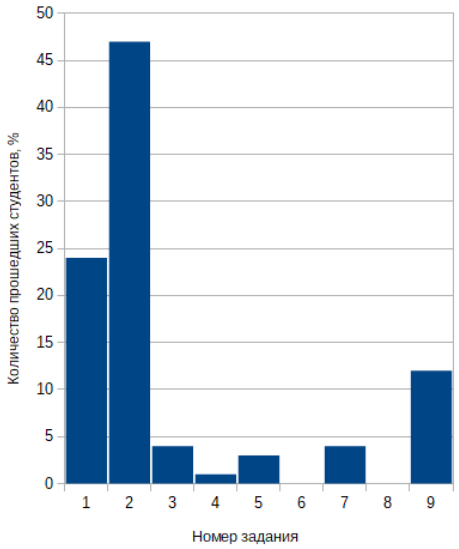

Рис. 2. Stopout

разница в сложности между второй и третьей задачей слишком велика, что оттолкнуло студентов от ее решения.

B. 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 оставил ту или иную задачу. На рис. 3 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, этот этап вынесен на отдельный график на рис. 4. Данные, представленные для каждой задачи на обоих графиках, высчитывались относительно только тех студентов, которые приступили к ее выполнению.

Из графика на рис. 4 видно, что с каждой новой задачей, все больше студентов, которые брались за ее выполнение, старались довести ее до конца. Из графика на рис. 3 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.

Поэтапное исследование решений задачи позволило понять, что в первых двух задачах курса, большинство студентов прошли только самый первый этап и закончили решение на неудавшейся сборке. 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.

C. Описание дополнительной задачи

В качестве промежуточной задачи, которую предлагается добавить между второй и третей, предпочтительно использовать задания, позволяющие студенту лучше познакомиться со структурой модуля ядра, представлением переменных в памяти и стандартными макросами.

В качестве одного из вариантов, по условию задачи студенту может быть предложена реализация одной из простейших структур данных, таких как стек, очередь или односвязный список. Студенту необходимо будет

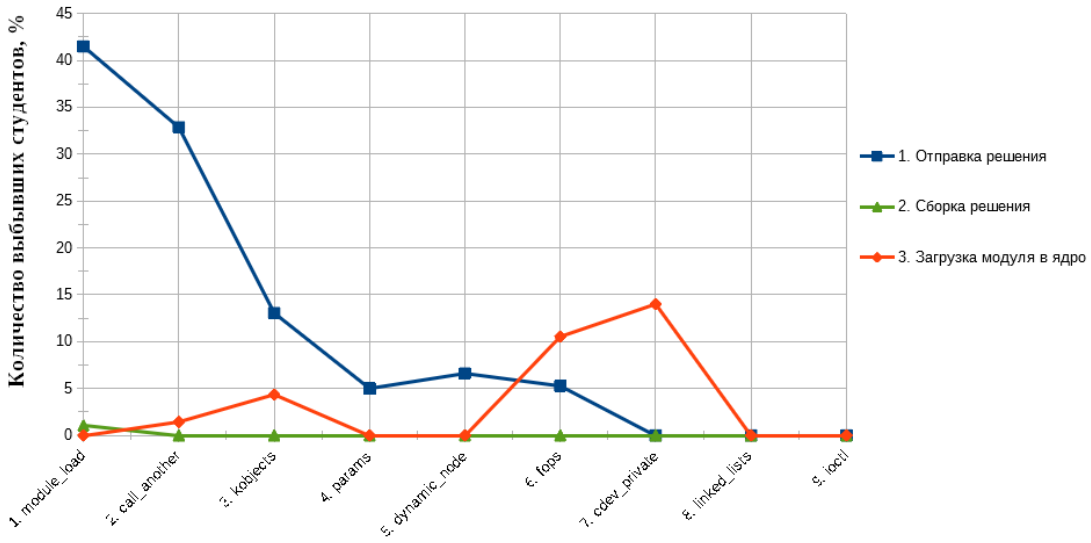

Рис. 3. Распределение студентов по пройденным ими этапами в рамках каждой задачи

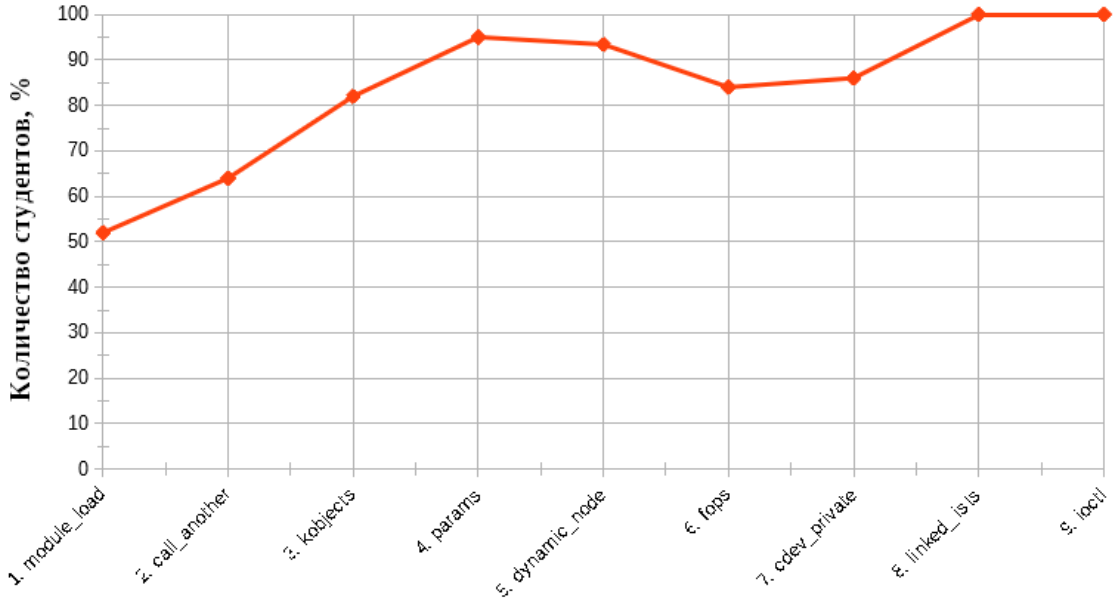

Рис. 4. Количество студентов, решивших задачу среди взявшихся за ее выполнение

реализовать свои функции добавления, удаления и получения элементов, а так же сделать их доступными из других модулей. Для автоматизации проверки данной задачи будет запущен модуль, взывающий функции, реализованные студентом, и проверяющий, что операции со структурой данных выполняются корректно.

Данная задача использует только навыки студента из предыдущих двух задач и его знание языка C. Студент при этом научится использованию макроса EXPORT_SYMBOL, написанию собственных функций и выделению памяти для переменных.

IV. 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, для того, чтобы облегчить понимание студентами условий последующих задач, тем самым уменьшив 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] Zaslavskiy, M. System for Automatic Checking of Student Solutions for Linux Programming MOOCs / M. Zaslavskiy, M. Kanushin // Proceedings of Software Engineering and Information management conference 2017. - 2017.

[5] Stepik, URL: https://stepik.org/ [Посещено: 13 Ноя 2017]

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

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

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