

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б10-мм

Рекомпиляция с помощью Ghidra: глобальные переменные

Сарапулов Василий Алексеевич

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
старший преподаватель кафедры информационно-аналитических систем, К. К. Смирнов

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	5
2. Обзор	6
2.1. Копирование данных из ассемблера	6
2.2. Глобальные переменные	6
2.3. Вывод значений в ассемблере	7
3. Реализация	8
3.1. Копирование данных	8
3.2. Экспорт глобальных переменных	9
3.3. Вывод значений в ассемблере	10
3.4. Конфликт типов данных	10
4. Эксперимент	11
4.1. Условия эксперимента	11
4.2. Исследовательские вопросы	11
4.3. Результаты	12
Заключение	13
Список литературы	14

Введение

Исследование и анализ исполняемых файлов становятся все более актуальными в области информационной безопасности и обратного проектирования программного обеспечения. Декомпиляция является важной частью этого процесса, поскольку позволяет преобразовать исполняемые файлы в человекочитаемый вид, что упрощает анализ [1]. Ghidra — это фреймворк для обратного инжиниринга программного обеспечения, включающий набор инструментов для анализа бинарных файлов [5].

Рекомпиляция отличается от декомпиляции тем, что полученный код может быть повторно скомпилирован, и полученный исполняемый файл будет делать то же самое, что и исходный. Рекомпиляция необходима для поднятия исходного кода программ, скомпилированных под одну архитектуру, и последующей компиляции на процессорах другой архитектуры. Поднятие кода — перевод кода, скомпилированного под конкретную архитектуру, в код высокоуровневого языка. Также рекомпиляция позволяет внести изменения в исходный исполняемый файл и заново его скомпилировать.

Хотя Ghidra предоставляет инструменты для декомпиляции, анализа исполняемого файла и обратного инжиниринга, полученный декомпилированный код не всегда может быть повторно скомпилирован. И так как Ghidra позиционируется как инструмент для анализа исполняемых файлов, а не для рекомпиляции, разработчики не ведут работу в этом направлении. Поэтому в полученном декомпилированном коде некоторые сложные синтаксические конструкции заменяются на более простые, человекочитаемые, но не соответствующие стандартам языка C.

Таким образом, для успешной рекомпиляции необходимо внести изменения и доработки в существующие методы анализа исполняемых файлов. Ранее в рамках летней школы 2024 был разработан постобработчик декомпилированного кода, который разрешал затруднения в декомпилированном коде. В прошлом семестре наша команда начала

переносить изменения, реализованные в постобработчике, в исходный код Ghidra. Была улучшена работа с заголовочными файлами, реализовано исключение служебных функций из декомпилированного кода и внесены улучшения в пользовательский интерфейс.

В рамках данной курсовой работы планируется реализовать два ключевых усовершенствования в Ghidra. Во-первых, будет реализован механизм работы с глобальными переменными, включая их корректный экспорт. Во-вторых, будет модифицирован вывод числовых значений в ассемблерном коде для соответствия синтаксису языка C, что улучшит читаемость и удобство работы с декомпилированным кодом.

1. Постановка задачи

Целью работы является разработка и реализация усовершенствований в Ghidra, направленных на достижение рекомпилируемости декомпилированного кода за счёт улучшения работы с глобальными переменными. Дополнительно в рамках работы планируется улучшение пользовательского интерфейса для повышения удобства анализа кода. Для её выполнения были поставлены следующие задачи:

1. Реализовать функцию копирования данных глобальных переменных из ассемблера в синтаксисе языка C с последующей интеграцией в пользовательский интерфейс.
2. Спроектировать и реализовать механизм экспорта глобальных переменных.
3. Изменить формат отображения числовых значений в ассемблерном представлении (переход от формата "ffh" к стандартному для C "0xff").
4. Провести сравнительный анализ рекомпилируемости кода путем измерения количества ошибок компиляции в исходной и модифицированной версиях Ghidra.

2. Обзор

2.1. Копирование данных из ассемблера

Для удобства анализа бинарных файлов в Ghidra предусмотрена функция "Copy Special", позволяющая копировать из ассемблерного кода данные в различных форматах (байтовые строки, массивы и другие)[4]. Однако текущая реализация не поддерживает копирование данных в виде объектов языка C, что ограничивает возможности работы с декомпилированным кодом. По данной проблеме есть issue[3] в репозитории Ghidra.



Рис. 1: Функция Copy Special

2.2. Глобальные переменные

Экспорт глобальных переменных играет ключевую роль в процессе рекомпиляции. В актуальной версии Ghidra 11.3.2 данная функциональность отсутствует в готовом виде, хотя соответствующая возможность реализована в исходном коде, о чем свидетельствует закрытое issue[2]. Однако текущая реализация решения данной проблемы предусматривает не полноценный экспорт, а лишь имитацию глобальных переменных. Конкретно, система формирует список таких переменных исклю-

чительно на основе их использования внутри функций. В результате при экспорте глобальные переменные получают неинициализированными, несмотря на наличие исходных инициализированных значений в анализируемом исполняемом файле.

Листинг 1: Пример исходного файла с инициализированными глобальными переменными

```
const char *str = "Hello, World!";
int int_var = 0x123456;
struct a structure = {1, 'b', {2, 4, 8}, "Hello", 0.5};
float float_var;

...
```

Листинг 2: Пример экспорта глобальных переменных при декомпиляции (Ghidra 11.5 Build DEV)

```
undefined main;
pointer __dso_handle;
undefined1 completed.0;
a structure;
int int_var;
char *str;
float float_var;
```

2.3. Вывод значений в ассемблере

Также одной из задач по данному проекту было приведение формата вывода числовых значений в ассемблерном коде к стандартному синтаксису языка C (0xff вместо ffh). Это изменение нацелено на упрощение анализа и дальнейшей работы с полученным кодом.

3. Реализация

В ходе работы были внесены модификации в исходный код Ghidra, направленные на устранение выявленных ограничений при работе с декомпилированным кодом. Также для реализованных функций были разработаны модульные тесты, обеспечивающие проверку корректности внесённых изменений. Основные доработки затронули следующие компоненты системы:

3.1. Копирование данных

Для повышения удобства работы с глобальными переменными была добавлена функция копирования данных в формате языка C. Пользователь может вызвать соответствующее действие через контекстное меню в окне декомпилятора, после чего в буфер обмена помещается корректное значение.



Рис. 2: Копирование структуры
(Скопированное значение: {0x1, 'b', {0x2, 0x4, 0x8}, "Hello", 0.5})

Для реализации данной функции в интерфейс Data был добавлен метод по умолчанию `getValueAsCObject`, обеспечивающий получение значения текущего элемента памяти в виде объекта языка C. Также были модифицированы методы `getValue` в `CompositeDB` и `getArrayValue`

в Аггау для корректного получения значений структур, объединений и массивов. Для интеграции в пользовательский интерфейс был добавлен метод `copyCObject` в класс `CodeBrowserClipboardProvider`, обеспечивающий копирование C-декларации через буфер обмена.

3.2. Экспорт глобальных переменных

Механизм экспорта глобальных переменных предназначен для автоматизированного формирования исходного кода на языке C с корректными объявлениями глобальных данных, извлечённых из бинарного файла. Эта функция упрощает интеграцию глобальных переменных в исходные проекты и повышает рекомпилируемость декомпилированного кода.

Для реализации функции экспорта глобальных переменных был модифицирован класс `CppExporter`, в который добавлен метод `writeProgramData`. Этот метод выполняет обход всех секций программы и обрабатывает только те из них, которые содержат данные [6]. В процессе обхода извлекаются переменные, расположенные в допустимых секциях, после чего для каждой переменной формируется C-декларация.

Дополнительно реализована фильтрация переменных с некорректными именами, не соответствующими синтаксису языка C, и исключение служебных переменных, что позволяет избежать ошибок при повторной компиляции. Такой подход обеспечивает генерацию корректного и пригодного к использованию исходного кода.

Листинг 3: Пример экспорта глобальных переменных при декомпиляции

```
char *s_Hello_00102012 = "Hello";
undefined4 DAT_00102024 = 0x3fc00000;
int int_var = 0x123456;
char *str = "Hello, World!";
a structure = {0x1, 'b', {0x2, 0x4, 0x8}, "Hello", 0.5};
float float_var;
```

3.3. Вывод значений в ассемблере

Также одной из задач по данному проекту было изменение формата вывода значений в ассемблере. Ранее Ghidra отображала шестнадцатеричные значения в формате FFh. В рамках данной модификации формат был изменён на 0xFF, что делает вывод более понятным и совместимым с синтаксисом C, упрощая анализ.

Для реализации изменения был модифицирован метод `getRepresentation` в классах, отвечающих за генерацию строкового представления чисел в ассемблерном представлении. Новая логика форматирования добавляет префикс 0x при отображении шестнадцатеричных значений.

```
int_var  
  
00104010 56 34      int      0x123456  
12 00
```

Рис. 3: Пример вывода значения в окне ассемблера

3.4. Конфликт типов данных

Также в рамках данной работы была исправлена ошибка, возникшая из-за изменений, сделанных в прошлом семестре. При автоматическом подключении заголовочных файлов возникал конфликт между типами данных, определёнными в этих файлах, и типами, автоматически сгенерированными Ghidra при экспорте. Одним из способов решения было отключение опции экспорта типов данных, однако это приводило к другим ошибкам, связанным с отсутствием необходимых типов, созданных в ходе декомпиляции. В текущем семестре был реализован механизм, позволяющий исключать из экспорта те типы данных, которые уже присутствуют в заголовочных файлах, тем самым устраняя конфликты и обеспечивая корректную работу декомпилированного кода.

4. Эксперимент

4.1. Условия эксперимента

Для проведения эксперимента был собран набор тестовых программ из домашних заданий по программированию на языке C, бенчмарков и системных утилит linux:

- `hello_world`. Программа, выводящая "hello World!".
- `bmp`¹. Программа чтения заголовков `bmp` файлов.
- `linpack`². Бенчмарк для измерения вычислительной производительности компьютеров при обработке чисел с плавающей запятой.
- `echo`. Системная утилита linux для вывода текста.
- `cat`. Системная утилита linux для вывода содержимого файла.

Компиляция первых трех программ проводилась с помощью компилятора `gcc` версии 14.2.0, с флагом `-g` для корректной выгрузки глобальных структур. Системные утилиты были взяты из пакета `coreutils` Ubuntu 25.04.

4.2. Исследовательские вопросы

Основной задачей данного эксперимента является проверка уменьшения количества ошибок компиляции при компиляции декомпилированного кода с автоматической выгрузкой глобальных переменных. Для сравнения результатов экспорт кода проводился трижды: без опций экспорта, с опциями, реализованными в прошлом семестре, с выгрузкой глобальных переменных.

Таблица 1: Сравнение количества ошибок компиляции при экспорте кода с разными опциями.

Программа	Без доп. опций экспорта	Результат прошлого семестра	Результат текущего семестра
hello_world	9	0	0
bmp	10	6	6
linpack	126	107	106
echo	213	142	142
cat	326	245	238

4.3. Результаты

Результаты эксперимента, представленные в таблице 1, показывают, что реализованные изменения не привели к существенному сокращению количества ошибок компиляции. Однако важно отметить, что корректный экспорт глобальных переменных играет критически важную роль для обеспечения правильности декомпилированного кода. Даже незначительное уменьшение количества ошибок свидетельствует о повышении качества выходного кода.

¹<https://github.com/Sarapulov-Vas/BMP>

²<https://github.com/ereyes01/linpack>

Заключение

В рамках данной учебной практики были достигнуты следующие результаты:

- Исправлена ошибка конфликта типов данных из заголовочных файлов и автоматически сгенерированных.³
- Реализована функция копирования данных из ассемблерного представления в формат языка C.
- Добавлена опция выгрузки глобальных переменных при экспорте кода на языке C.⁴
- Изменён формат вывода значений в ассемблере на стиль, соответствующий синтаксису языка C.⁵

³<https://github.com/spbu-se/ghidra/pull/10>

⁴<https://github.com/spbu-se/ghidra/pull/8>

⁵<https://github.com/spbu-se/ghidra/pull/9>

Список литературы

- [1] Cifuentes Cristina, Gough K. John. Decompilation of binary programs // [Software: Practice and Experience](#). — 1995. — Vol. 25, no. 7. — P. 811–829. — <https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.4380250706>.
- [2] Issue 2635 Export data as C code. — URL: <https://github.com/NationalSecurityAgency/ghidra/issues/2635> (дата обращения: 10 мая 2025 г.).
- [3] Issue 4668 Honor data type in special data copy. — URL: <https://github.com/NationalSecurityAgency/ghidra/issues/4668> (дата обращения: 10 мая 2025 г.).
- [4] NSA. Ghidra: Copy Special Documentation. — URL: <https://github.com/NationalSecurityAgency/ghidra/blob/82baf0aa747b63e2a642f90d62a222cb2c30f8d8/Ghidra/Features/Base/src/main/help/help/topics/ClipboardPlugin/Clipboard.htm#L18> (дата обращения: 13 июня 2025 г.).
- [5] NSA. Описание Ghidra. — URL: <https://github.com/NationalSecurityAgency/ghidra/blob/master/README.md> (дата обращения: 20 мая 2025 г.).
- [6] elf(5) File Formats Manual, 2024. — URL: <https://man7.org/linux/man-pages/man5/elf.5.html> (дата обращения: 4 мая 2025 г.).