Информационный проект «оптимизация фильтрации текстовых данных с помощью низкоуровневого программировния в среде GNU/Linux»

Гордиенков Захар Юрьевич

22 октября 2023 г.

Содержание

1 Вступление **4**

2 Концепция низкоуровнего программирования **5**
2.1 История 5
2.2 Современность 5

3 Особенности языка Ассемблера **5**
3.1 Команды 5
3.2 Процесс сборки исполняемого файла 5

4 Отчёт о работе над продуктом **5**

5 Вывод **6**

6 Список литературы **7**

7 Приложения **8**
7.1 Глоссарий 8

Цель: создать средство фильтрации текстовых данных (в т.ч. системных логов), которое будет превосходить уже существующие решения в производительности

Задачи:

- Изучить базовые концепции низкоуровневого программирования
- Изучить синтаксис и особенности языка ассемблера
- Рассмотреть доступные средства фильтрации текстовых данных в системах GNU/Linux
- Создать свою более производительную систему фильтров

1 Вступление

Малая эффективность средств текстовой фильтрации является актуальной проблемой в сфере системного администрирования. В частности, сложность использования, отсутствие гибкости и низкая скорость работы. Считается, что при использовании языков низкого уровня можно оптимизировать как использование системных ресурсов, так и сократить время выполнения программы за счёт прямого доступа к аппаратным компонентам системы и возможности оптимизации под определённую системную архитектуру. Следовательно, можно предположить, что с помощью низкоуровнего программирования можно решить проблему низкой скорости работы при фильтрации текстовых данных. Такое ПО могло бы быть востребованным.

В рамках этого проекта изучался ассемблер NASM [1], который работает на аппаратной архитектуре x86 16-и, 32-х и 64-битной разрядности. Ассемблер до сих пор поддерживается разработчиками, имеет избыточную документацию, является кроссплатформенным, поэтому при выборе инструментов для создания продукта было выбрано именно это ПО. В рамках этого проекта для простоты рассматривался только набор инструкций, поддерживаемый 32-битной архитектурой.

[1] NASM (Netwide Assembler) 2.16.01, автором которого является британский программист Саймон Тэтхем (Simon Tatham), распространяется по упрощённой лицензии BSD (Simplified (2-clause) BSD License)

2 Концепция низкоуровнего программирования

2.1 История

2.2 Современность

3 Особенности языка Ассемблера

3.1 Команды

3.2 Процесс сборки исполняемого файла

4 Отчёт о работе над продуктом

5 Вывод

6 Список литературы

- Э. Немет, Г. Снайдер, Т. Хейн, Б. Уэйли, Д. Макин – «Unix и Linux: руководство системного администратора, 5-е издание »; СПб: ООО ‘Диалектика‘, 2018. 1168 страниц
- Зубков С. В. – «Assembler. Для DOS, Windows и Unix »; М: ДМК Пресс, 2017. 638 страниц
- Robert C. Martin – «Clean Architecture. A Craftsman's Guide to Software Structure and Design »; Pearson Education Inc, 2018. 350 страниц
- Документ «Intel® 64 and IA-32 Architectures: Software Developer's Manual »; Май 2007
- Документ « INTEL 80386. PROGRAMMER'S REFERENCE MANUAL»; 1986. 421 страница

Искренне благодарен авторам вышеперечисленных книг, статей и видеоматериалов.

7 Приложения

7.1 Глоссарий

- системы GNU/Linux (также просто Linux) – семейство свободных Unix-подобных операционных систем, основанное на утилитах операционной системы GNU, разработанной Ричардом Мэтью Столлманом и ядре Linux, создателем которой является Линус Торвальдс

- Ассемблер – (не путать с языком ассемблера, который часто называют просто ассемблером) - транслятор программы из текста на языке ассемблера в программу на машинном языке (объектный файл). Самые известные ассемблеры на данный момент – TASM, FASM, NASM и MASM. Не путать с языком ассемблера, который зачастую называют ассемблером

- Ассемблирование – процесс работы ассемблера

- Язык ассемблера (англ. assmebly language) – представление команд процессора в понятном для человека виде. Большинство ассемблеров имеют свои синтаксические особенности при записи процессорных команд. Не путать с ассемблером

- Компоновщик (редактор связей, англ. linker) – прикладная программа, которая собирает исполняемый файл из одного или нескольких объектных модулей

- Исполняемый файл – машинный код, который предназначен для чтения и выполнения процессором. В Unix-системах для таких файлов используется формат ELF (см. ниже), в Windows - PE (Portable Executable)

- Регулярные выражения (англ. regular expressions) – формальный язык, используемый в программах, работающих с текстовыми данными, для поиска подстрок и манипуляций с ними, в основе которого - использование символов подстановки (символ-джокер, англ. wildcard character). Строка, по которой осуществляется поиск, называется маской (шаблон, англ. pattern)

- Объектный файл – файл, содержащий представление части программы. Предназначен для соединения с другими объектными

файлами в исполнимый модуль (или библиотеку) с помощью компоновщика. Объектные файлы в Linux имеют формат ELF, в Windows - COFF

- ELF, Executable & Linlable Format – формат исполнимых и компонуемых файлов, используемый во многих Unix-подобных системах. ELF-64 – формат ELF, адаптируемый под 64-разрядную архитектуру
- Unix, Unix-подобные системы – группа многопользовательских многозадачных переносимых операционных систем, в основе которых лежит одноимённый проект компании AT&T.
- Intel-синтаксис – формат записи инструкций процессора, используемый в документации к продуктам компании Intel.
- Разрядность процессора – размер (в битах) машинного слова в таком процессоре
- Лицензия BSD (Berkeley Software Distribution license) – программная лицензия университета Беркли. Этот документ впервые был применён при распространении UNIX-подобных операционных систем BSD
- x86 – архитектура процессора и набор команд, созданный компанией Intel
- Архитектура процессора – структурное устройство процессора. Процессоры одной архитектуры устроены схожим образом и имеют одинаковый набор машинных команд. Например, ARM (часто встречается среди мобильных устройств), RISC-V (экспериментальный проект, активно развивающийся в последние годы), x86-64 (характерна для современных персональных компьютеров).