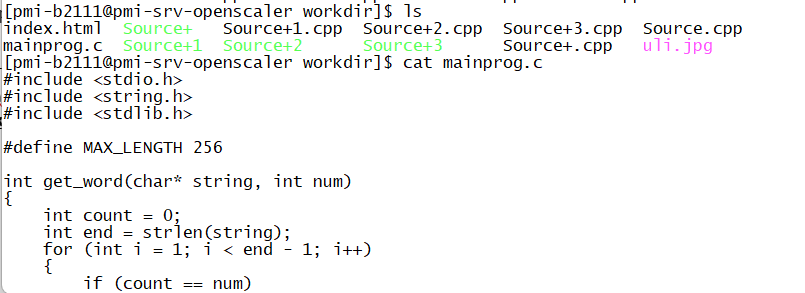
|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 6 | | |
| по дисциплине «Операционные системы, среды и оболочки» | | |
| **Инструментальные средства разработки программ** | | |
|  | | |
|  | Бригада 11 | Егупов Иван |
| Группа ПМ-04 | Порсин данил |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | Кобылянский В. Г. |
|  | сивак м.а. |
| Новосибирск, 2024 | | |

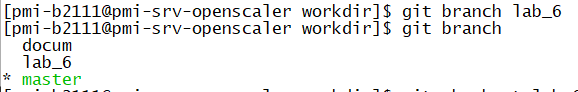
1. **Цель**

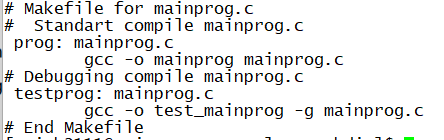
Целью работы является изучение основных этапов разработки и отладки приложений в ОС Linux, а также приобретение практических навыков по использованию инструментальных средств фонда свободного программного обеспечения при компиляции исходного кода, сборке, отладке и тестировании программ, написанных на языке Си.

1. **Ход работы**
2. Выберите из Приложения 2 программу в соответствии с номером бригады и скопируйте ее в файл **~/workdir/mainprog.c**

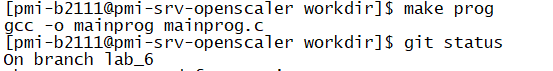
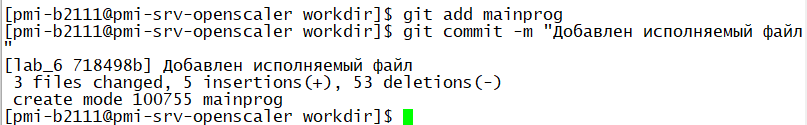
**Программа 11. Реверсирование каждого из пары слов**

2)С помощью редактора **vi** в рабочем каталоге сервера создайте make-файл, а в локальном репозитории создайте новую ветку *lab\_6*. Основные сведения по редактору приведены в приложении 1.



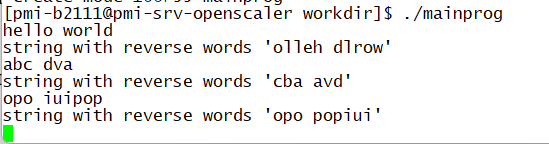


3)С помощью make-файла выполните компиляцию файла **mainprog.c**, используя правило ***prog***. Обнаруженные при компиляции ошибки исправьте с помощью редактора **vi**. После каждого исправления измененную версию программы сохраните в ветке *lab\_6* с поясняющим комментарием. Сведения об ошибках занесите в отчет (номер строки, значение строки до устранения и после устранения ошибки и пояснения).

Ошибок при компиляции не обнаружено.

4)Запустите исполняемый файл **mainprog** и проверьте программу на подготовленных наборах тестовых данных, результаты тестирования анесите в отчет.



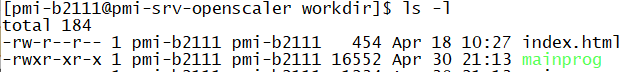
5)При обнаружении семантических ошибок перекомпилируйте программу с помощью правила ***testprog*** make-файла и запустите отладчик **gdb**, с помощью которого найдите причины появления ошибок. Проверьте программу на всех наборах тестовых данных. Сведения о найденных ошибках занесите в отчет, исправленную версию программы сохраните в репозитории с поясняющим комментарием.

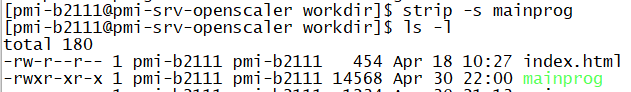
На наборе тестовых данных семантических ошибок не обнаружено. Программа работает исправно

6)Выведите журнал всех изменений файла **mainprog.c**, выполненных в ходе отладки программы, занесите список в отчет.

Изменений файла не проводилось

7)Определите размер исполняемого модуля отлаженной программы. Удалите всю отладочную информацию и снова определите размер исполняемого модуля, сравните с предыдущим результатом, результат сравнения занесите в отчет и подтвердите скриншотом.

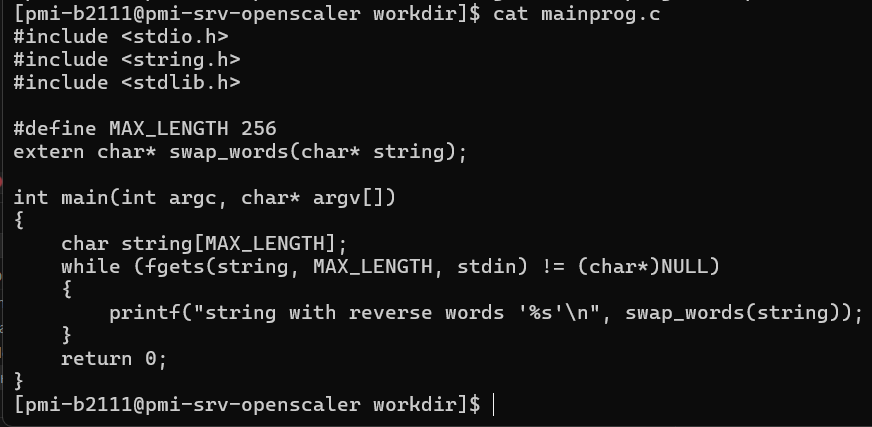


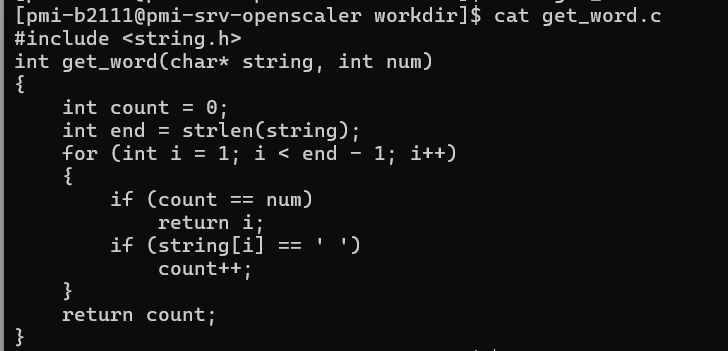


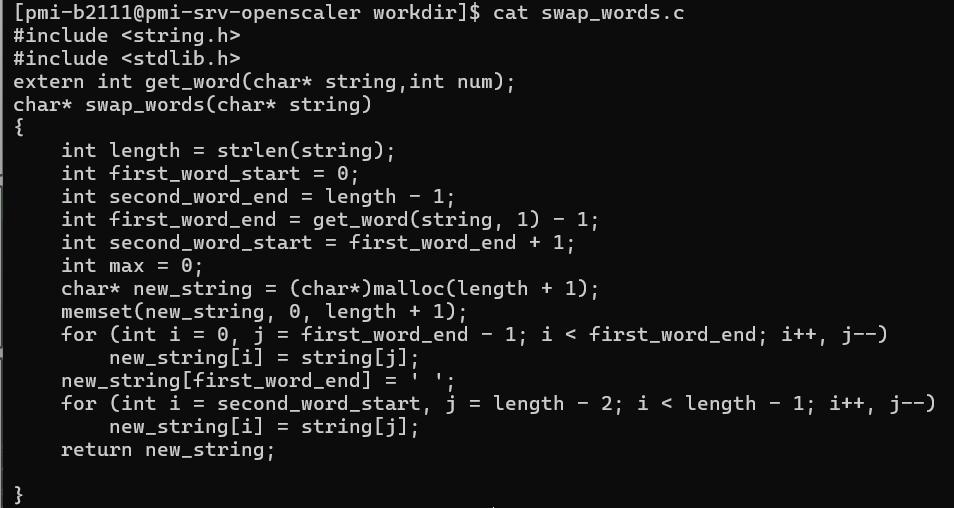
Размер исполняемого файла незначительно уменьшился (16552/14568)

8)Выполните разбиение программы **mainprog** на функции в соответствии с номером бригады из таблицы 3. Здесь prog1() и prog2() – условные имена функций, используемых в программе. Обратите внимание на тип функции (внутренняя или внешняя), тип файла (.c, .h или .o) и тип модуля (исходный или объектный). Занесите в отчет измененный текст программы.







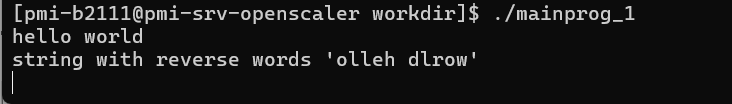






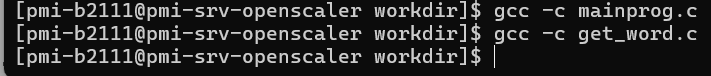
9)Выполните сборку программы в соответствии с вариантом задания (см. табл.3), используя неявный вызов компоновщика и задав имя исполняемого файла **mainprog\_1,** проверьте корректность работы программы и занесите в отчет результаты ее тестирования.

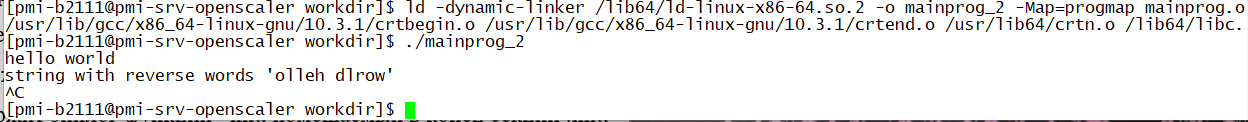




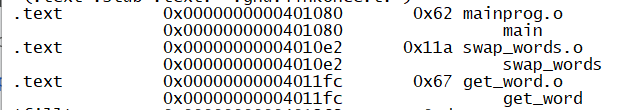
10)Выполните сборку программы в соответствии с вариантом задания (см. табл.3), используя явный вызов компоновщика. Результатом сборки должны быть исполняемый файл **mainprog\_2** и карта памяти **progmap;** проверьте корректность работы программы и занесите в отчет результаты ее тестирования.

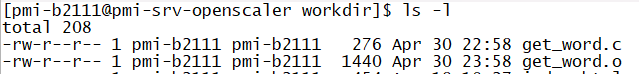
Для явного вызова компоновщика необходимо предварительно сформировать объектные модули mainprog.o и get\_word.o





11)Из карты памяти **progmap** определите размеры машинного кода модулей **mainprog.o,** **prog1.o** и **prog2.o,** сравните их с размерами исходного и объектного кода этих модулей. Размеры файлов типа **.с** и **.о** определите с помощью команды ls. Результат представьте в виде таблицы 4, все данные должны быть подтверждены скриншотами.









Машинный код в байтах записан в 16-ой системе счисления. Переведём эти значения:

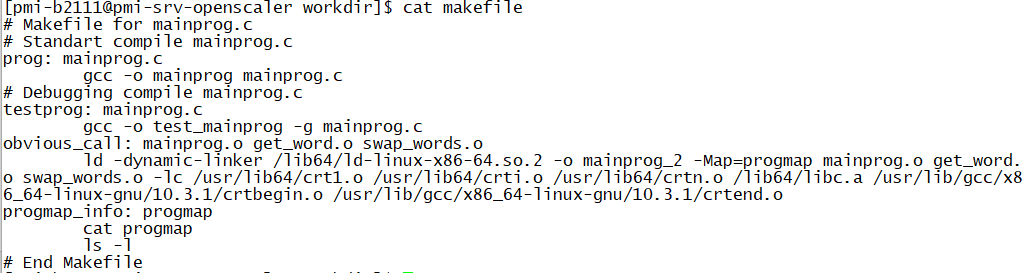
0x62=98

0x11a=282

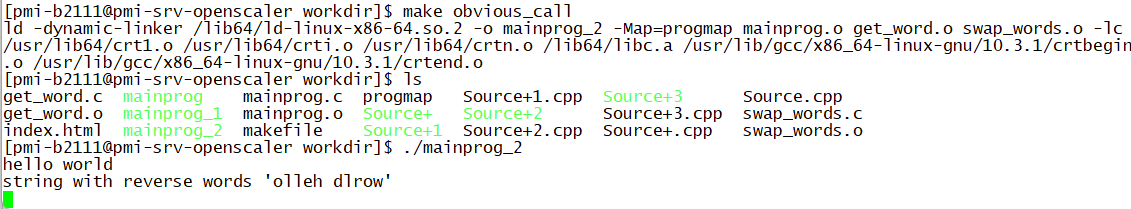
0x67=103

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля (функции) | Исходный, байт | Объектный, байт | Машинный код, байт |
| get\_word | 276 | 1440 | 103 |
| Swap\_words | 707 | 1784 | 282 |
| Mainprog | 363 | 1752 | 98 |

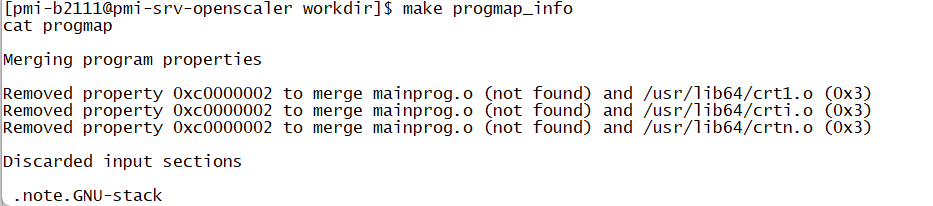
12)Добавьте в make-файл, разработанный при выполнении п. 2, два новых правила, реализующие п. 10 и 11 задания. Проверьте корректность его работы и сохраните его в репозитории.

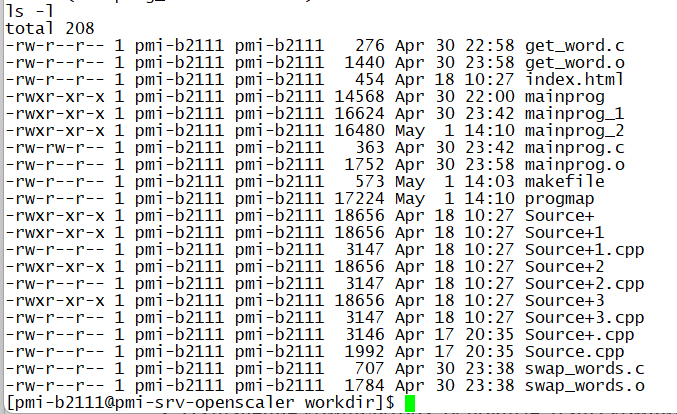


**Пункт 10:**

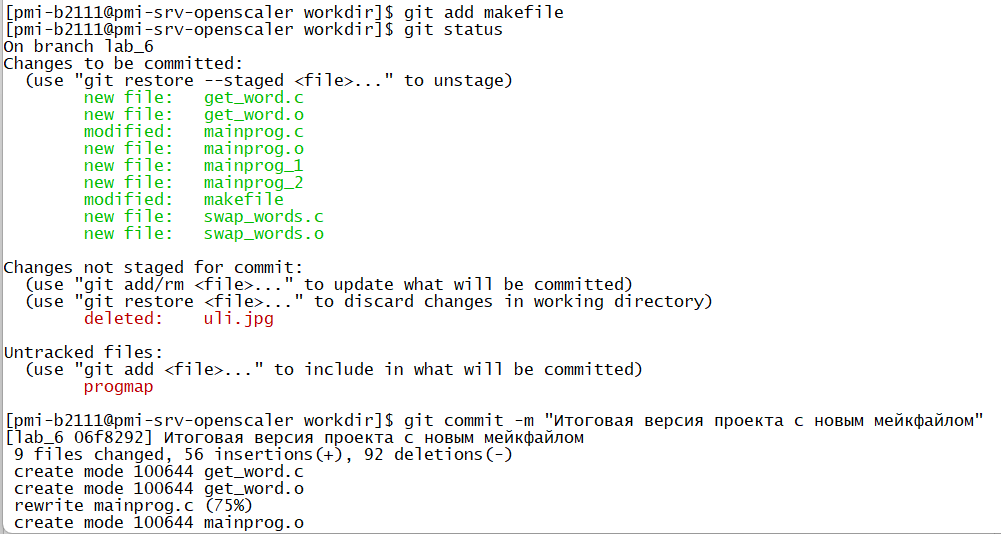


**Пункт 11:**





Добавление в репозиторий:



13)Выполните экспорт всех данных из локального репозитория в облачный репозиторий НГТУ.

