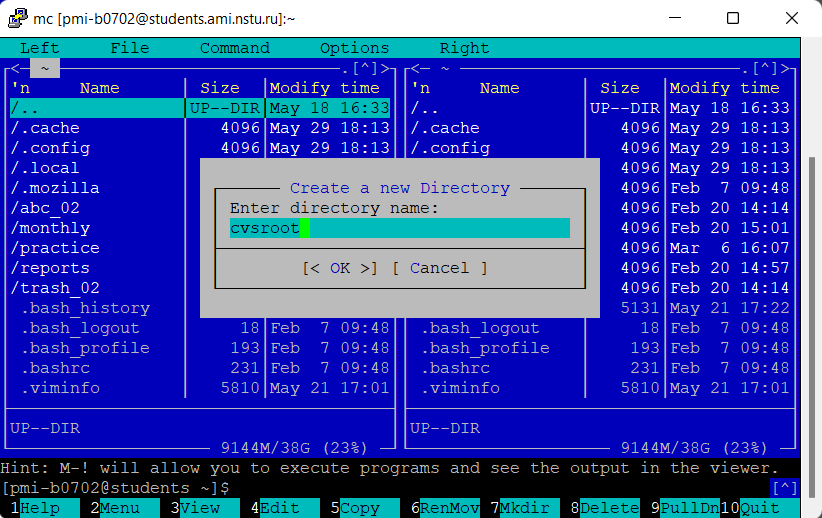
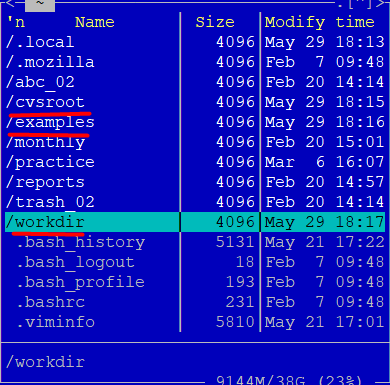
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Теоретической и прикладной математики | | |
|  | | |
| Лабораторная работа № 6 | | |
| по дисциплине «Операционные системы, среды и оболочки» | | |
|  | | |
| **Инструментальные средства разработки программ** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 2 |
| Студент: | Сидоров Даниил Игоревич |
|  |  |
| Преподаватель: | Кобылянский Валерий Григорьевич, |
|  | Филиппова Елена Владимировна  . |
|
|  |  |
|  | | |
| Новосибирск | | |
| 2022 | | |

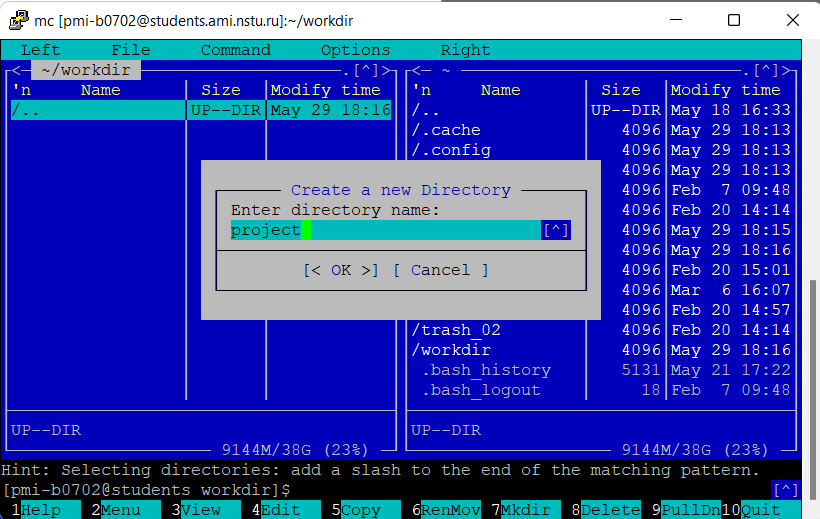
1. **Цель работы**

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. Запустили файловый менеджер **mc** и создали в домашнем каталоге подкаталоги **cvsroot,** **workdir** и **examples,** в каталоге **workdir** создали подкаталог **project.**

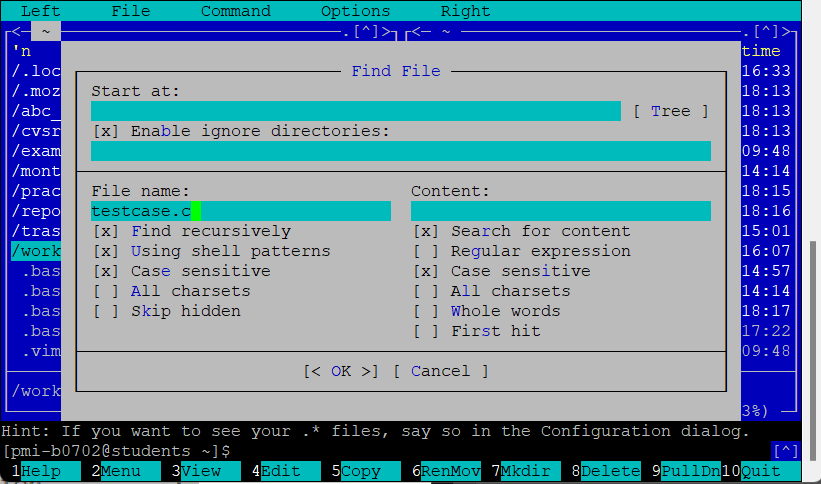


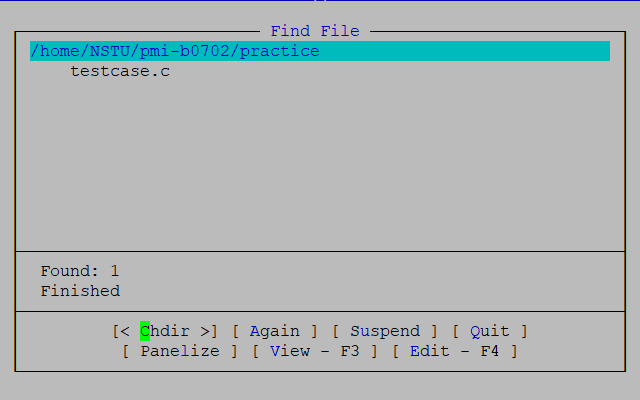




1. Выполнили поиск во внешней памяти сервера файла **testcase.c**

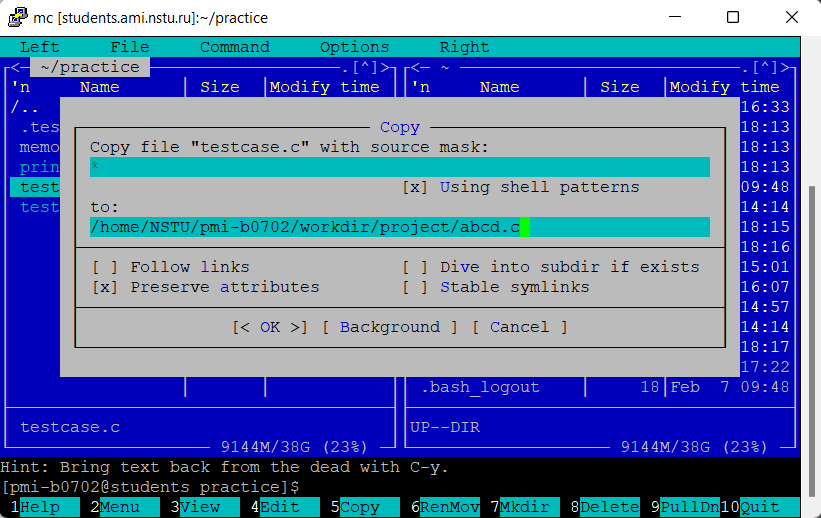
F9->Command->Find file





1. Скопировали файл **testcase.c** в каталог **project** под именем **abcd.c** и занесли в отчет исходный текст программы, предварительно сравнив его с текстом, приведенным в п.2.6.

Скопируем при помощи F5.



Исходный код:

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define WORD 0

#define IGNORE 1

/\* Globals, used by both subroutines. \*/

char \*Words[BUFSIZ/2]; /\* Worst case, single letters. \*/

int WordCount;

/\* Walk through the array of works, find those with the

\* matching charakter, printing them on stdout. Note that

\* the NULL charakter will match all words. \*/

void PrintWords(wc, match)

int wc; /\* Number of words in Words[] \*/

char match; /\* Attempt to match this charakter. \*/

{ register int ix; /\* Index in Words[]. \*/

register char \*cp; /\* Pointer for searching. \*/

for (ix=0; ix < wc; ix++) {

cp = Words[ix];

/\* Try to match the given character.

\* Scan the word, attempting to match,

\* or until the end of the word is found. \*/

while ((\*cp) && (\*cp++ != match));

if (\*cp == match) /\* Found a match? Write the word on stdout. \*/

(void) printf("%s0, Words[ix]); } return; }

/\* Find words in the gives buffer. The Words[] array is set

\* to point at words in the buffer, and the buffer modifeid

\* with NULL characters to delimit the words. \*/

int GetWords (buf)

char buf[]; /\* The output buffer. \*/

{ register char \*cp; /\* Pointer for scanning. \*/

int end = strlen(buf); /\* length of the buffer. \*/

register int wc = 0; /\* Number of words found. \*/

int state = IGNORE; /\* Current state. \*/

/\* For each character in the buffer. \*/

for (cp = &buf[0]; cp < &buf[end]; cp++) {

/\* A simple state machine to process

\* the current character in the buffer. \*/

switch(state) {

case IGNORE:

if (!isspace(\*cp)) {

Words[wc++] = cp; /\* Just started a word? Save it. \*/

state = WORD; /\* set the state. \*/ } break;

case WORD:

if (isspace(\*cp)) {

\*cp = '\0'; /\* Just completed aword? terminate it. \*/

state = IGNORE; /\* set the state. \*/ } break; }}

return wc; /\* Return the word count. \*/ }

int main(argc, argv) int argc; char \*argv[]; { char buf[BUFSIZ], match;

/\* Check command line arguments. \*/

if (argc < 2) match = ' ';

/\* No command line argument, match all words. \*/

else match = \*++argv[1]; /\* match the char after the first - \*/

/\* Until no more output on stdin. \*/

while(gets(buf) != (char \*)NULL) {

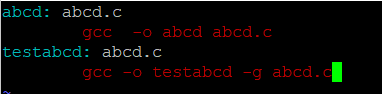
WordCount = GetWords(buf); /\* Paste the output buffer. \*/

PrintWords(WordCount, match); /\* Print the matching words. \*/ }

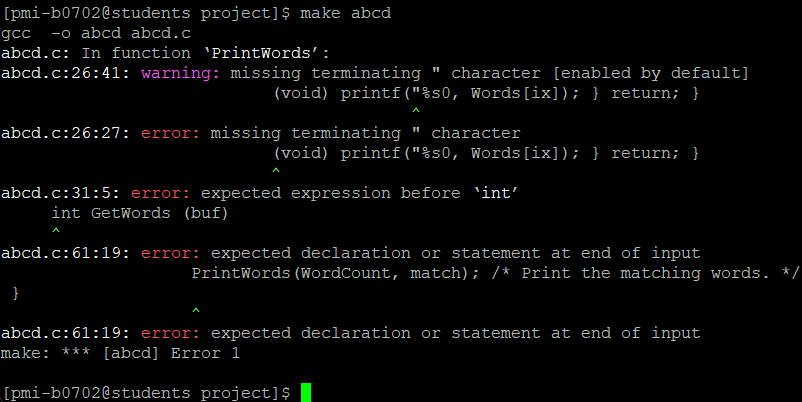
/\* Manifests for state machine to parse output line. \*/

1. С помощью редактора **vi** создали в каталоге **project** make-файл согласно п. 2.3.





1. Выполнили компиляцию программы **abcd.c** с помощью make-файла, используя правило **abcd**. Исправили синтаксические ошибки с помощью редактора **vi,** информацию по ошибкам и их устранению занесли в отчет (номер строки, значение строки до устранения и после устранения ошибки, пояснения).





**Строка №26**. Не закрыты кавычки. Заменили “%s**0** на “%s\n”

До исправления: 

После исправления: 

**Строка №61**. Нет строки “return(0);}”.

До исправления:

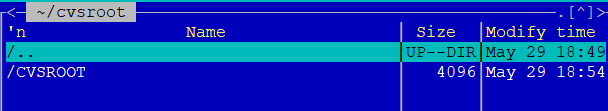


После исправления:



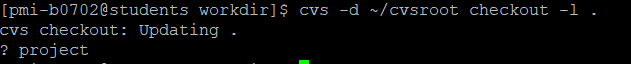
1. В каталоге **cvsroot** создали репозиторий CVS.





1. Передали каталог **project** и файл **abcd.c** в репозиторий. При выполнении команды **commit** с помощью редактора **vi** ввели комментарий, например: “Пользователь pmi-b0702 передал файл abcd.c под управление CVS”.

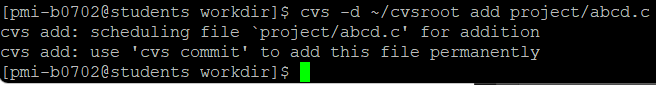
**Свяжем рабочий каталог с хранилищем:**



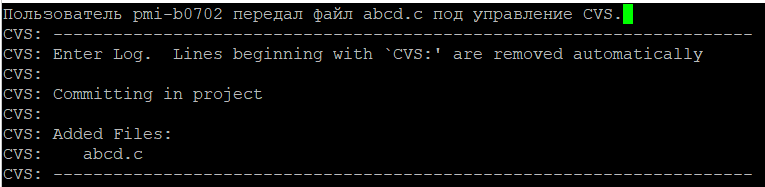
**Передадим проект в хранилище:**

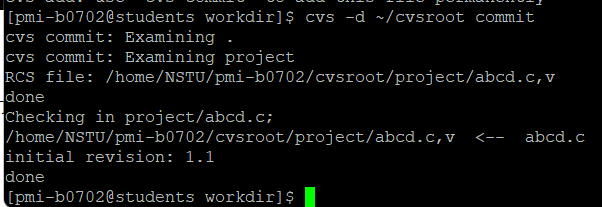


**Передадим файл abcd.c в хранилище:**

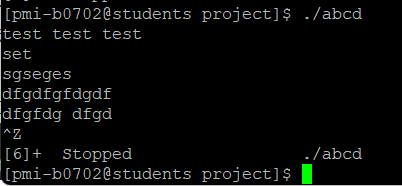


**Внесём изменения в репозиторий:**



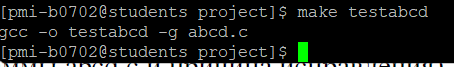


1. Запустили исполняемый файл abcd и пояснили результат запуска.

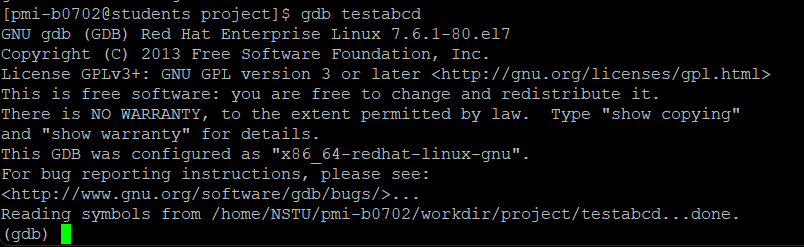


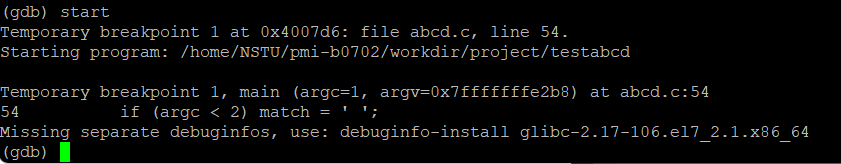
Запустили исполняемый файл abcd: ожидает ввода строки, но не работает после ввода символов. Программа работает неверно.

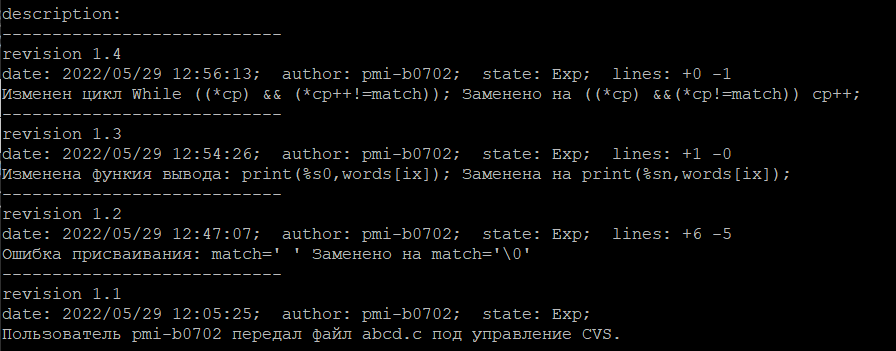
1. Перекомпилировали программу с помощью правила **testabcd** make-файла.

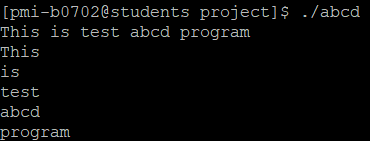


1. С помощью отладчика **gdb** выполнили поиск и устранение семантических ошибок в программе **abcd**. Каждое исправление в программе должно сопровождаться записью в репозиторий новой версии с комментарием, поясняющим на русском языке сущность исправлений (например, номер строки программы **abcd.c** и причина исправления). После устранения всех ошибок занесли в отчет результаты тестирования программы в двух вариантах запуска – без параметра и с параметром.

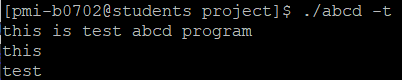








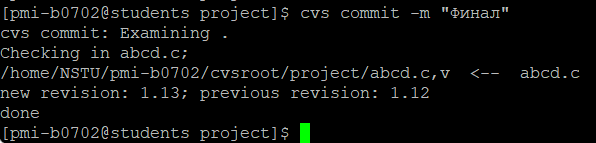
C параметром -t:



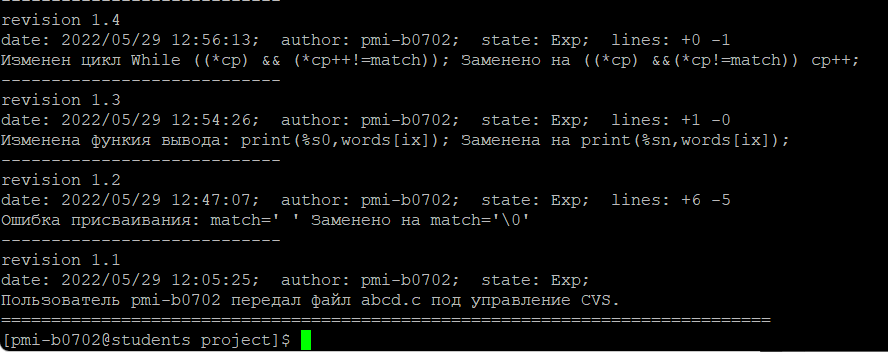
Программа работает верно.

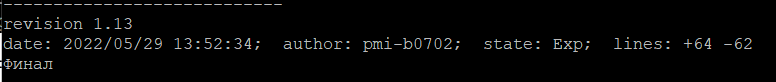
1. После получения корректных результатов выполнения программы **abcd** с помощью редактора **vi** в начало отлаженной программы ввели комментарий: "Программа abcd отлажена с помощью отладчика gdb дд.мм.гг. бригадой группы ПМИ-ХХ в составе: ФИО1, ФИО2..." и сохранили в репозиторий финальную версию программы.





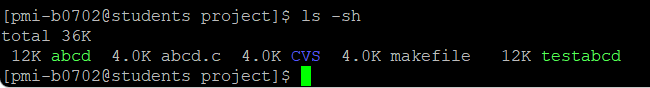
1. Вывели список изменений файла **abcd.c**, выполненных в ходе отладки программы, занесли список в отчет.





1. Определили размер исполняемого модуля отлаженной программы. Удалили всю отладочную информацию и снова определили размер исполняемого модуля, сравнили с предыдущим результатом, результат сравнения занесли в отчет.

Размер отлаженной программы

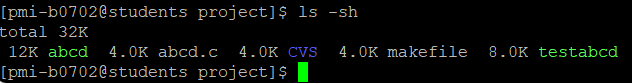


12Кб

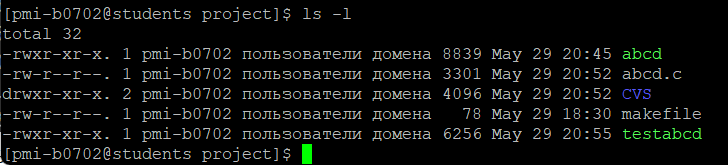
Удаление отладочной информации



Размер модуля после удаления отладочной информации



8кб

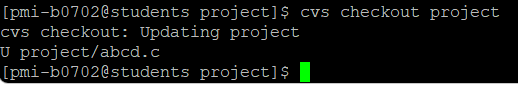


Размер исполняемого модуля отлаженной программы 6256 байт

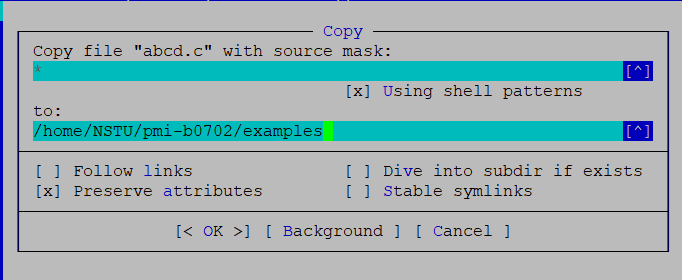
Размер Makefile 78 байт

1. Извлекли из репозитория полностью отлаженную программу **abcd** и скопировали её в каталог **example**, заменив в нем предыдущую версию программы. Все дальнейшие действия будут выполняться в этом каталоге.

Извлекли программу из репозитория



Копируем при помощи F5 в каталог examples



1. Выполнили разбиение полностью отлаженной программы **abcd** на функции в соответствии с номером бригады из таблицы 3. Занесли в отчет измененный текст программы.



**Abcd2.c:**

/\*Программа abcd отлажена с помощью отладчика gdb 29.05.2022 ПМИ-02 бригада №2:Сидоров Д.И. \*/

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include “printwords.h”

#define WORD 0

#define IGNORE 1

/\* Globals, used by both subroutines. \*/

char \*Words[BUFSIZ/2]; /\* Worst case, single letters. \*/

int WordCount;

int GetWords (buf)

char buf[]; /\* The output buffer. \*/

{ register char \*cp; /\* Pointer for scanning. \*/

int end = strlen(buf); /\* length of the buffer. \*/

register int wc = 0; /\* Number of words found. \*/

int state = IGNORE; /\* Current state. \*/

/\* For each character in the buffer. \*/

for (cp = &buf[0]; cp < &buf[end]; cp++) {

/\* A simple state machine to process

\* the current character in the buffer. \*/

switch(state) {

case IGNORE:

if (!isspace(\*cp)) {

Words[wc++] = cp; /\* Just started a word? Save it. \*/

state = WORD; /\* set the state. \*/ } break;

case WORD:

if (isspace(\*cp)) {

\*cp = '\0'; /\* Just completed aword? terminate it. \*/

state = IGNORE; /\* set the state. \*/ } break; }}

return wc; /\* Return the word count. \*/ }

int main(argc, argv) int argc; char \*argv[]; { char buf[BUFSIZ], match;

/\* Check command line arguments. \*/

if (argc < 2) match ='\0';

/\* No command line argument, match all words. \*/

else match = \*++argv[1]; /\* match the char after the first - \*/

/\* Until no more output on stdin. \*/

while(fgets(buf,sizeof(buf),stdin) != (char \*)NULL) {

WordCount = GetWords(buf); /\* Paste the output buffer. \*/

PrintWords(WordCount, match); /\* Print the matching words. \*/}

return(0);}

/\* Manifests for state machine to parse output line. \*/

**Printwords.h:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define WORD 0

#define IGNORE 1

/\* Globals, used by both subroutines. \*/

char \*Words[BUFSIZ/2]; /\* Worst case, single letters. \*/

int WordCount;

void PrintWords(wc, match)

int wc; /\* Number of words in Words[] \*/

char match; /\* Attempt to match this charakter. \*/

{ register int ix; /\* Index in Words[]. \*/

register char \*cp; /\* Pointer for searching. \*/

for (ix=0; ix < wc; ix++) {

cp = Words[ix];

/\* Try to match the given character.

\* Scan the word, attempting to match,

\* or until the end of the word is found. \*/

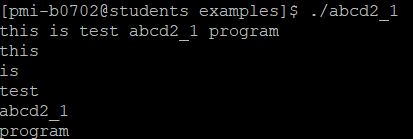
while ((\*cp) && (\*cp != match)) cp++;

if (\*cp == match) /\* Found a match? Write the word on stdout. \*/

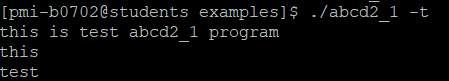
(void) printf("%s\n", Words[ix]); } return; }

1. Выполнили сборку программы в соответствии вариантом задания, используя неявный вызов компоновщика и задав имя исполняемого файла **adcd2\_1;** проверили корректность работы программы и занесли в отчет результаты ее тестирования.





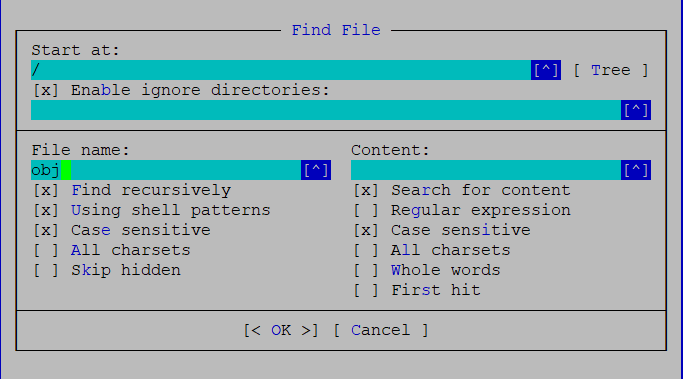
Тест с параметром -t:



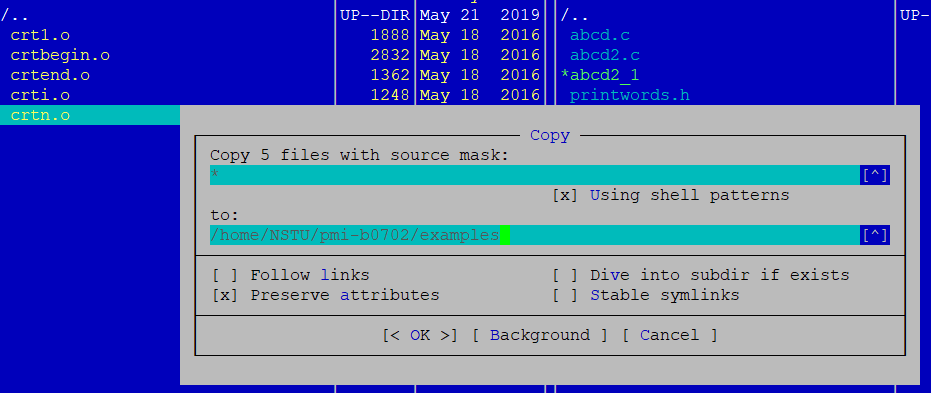
Программа работает верно

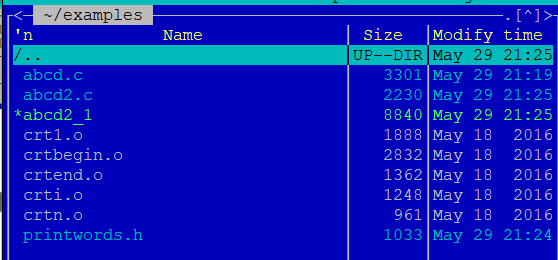
1. Выполнили поиск во внешней памяти сервера каталога **obj**, скопировали из него все файлы в каталог **example**, пояснили в отчете назначение скопированных файлов.

F9->Command->Find file



Выделяем все файлы и копируем при помощи F5.





**crt1.o** – содержит код, инициализирующий среду исполнения языка C и вызывающий определенную пользователем функцию main;

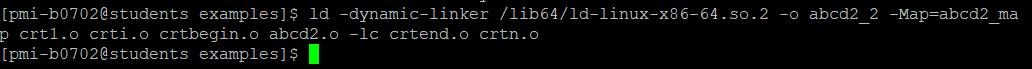
**сrti.o** – содержит пролог функции \_init, помещаемый в начало секции .init;

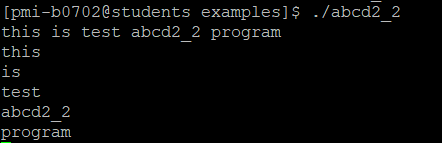
**crtn.o** – содержит эпилог функции \_init, помещаемый в конец секции .init;

**crtbegin.o, crtend.o** – обрабатывают глобальные конструкторы и деструкторы языка C++;

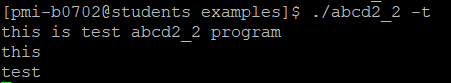
1. Выполнили сборку программы в соответствии вариантом задания, используя явный вызов компоновщика. Результатом сборки должны быть исполняемый файл **abcd2\_2** и карта памяти **abcd2\_map;** проверили корректность работы программы и занесли в отчет результаты ее тестирования.







Тест с параметром -t:



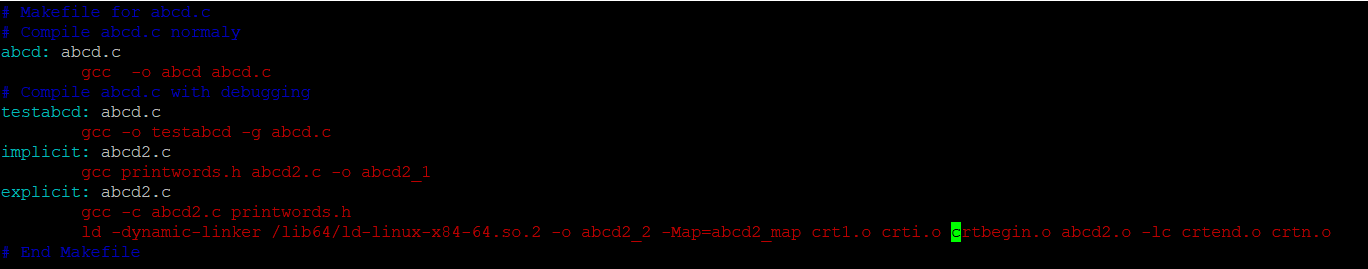
Все работает верно.

1. Из карты памяти **abcd2\_map** определили размеры машинного кода модулей **abcd2.o,** **printwords.o** и **getwords.o,** сравнили их с размерами исходного и объектного кода этих модулей (файлы типа **.с** и **.о)**. Результат представили в виде таблицы, все данные должны быть подтверждены скриншотами.

Открыли файл abcd2\_map c помощью F3

|  |  |  |  |
| --- | --- | --- | --- |
| Имя модуля (функции) | Исходный, байт | Объектный, байт | Машинный код, байт |
| printwords |  |  |  |
| getwords |  |  |  |
| abcd2 |  |  | 481 байт |

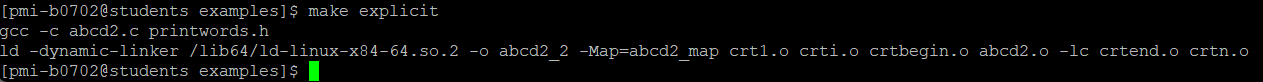
1. Добавили в make-файл, разработанный при выполнении п.5, два новых правила, реализующие п. 16 и 18 задания. Проверили корректность его работы.



Добавили правило implict для сборки программы, используя неявный вызов компоновщика и правило explicit для сборки программы, используя явный вызов компоновщика.









Работает корректно.

1. **Вывод**

Контрольные вопросы проработаны.