Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра теоретической прикладной информатики

Лабораторная работа № 6
по дисциплине «Операционные системы, среды и оболочки»

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПРОГРАММ

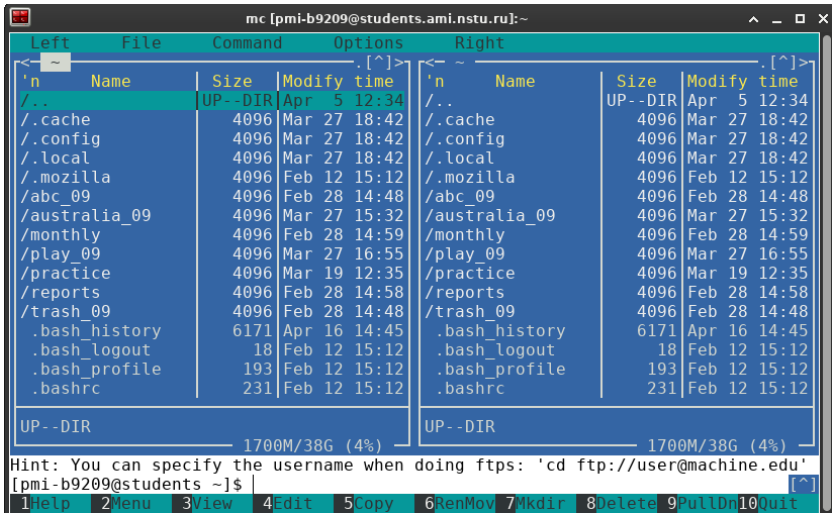
Факультет:	ПМИ
Группа:	ПМ-92
Бригада:	9
Студенты:	Иванов В., Кутузов И.
Преподаватель:	Сивак М.А.

Новосибирск

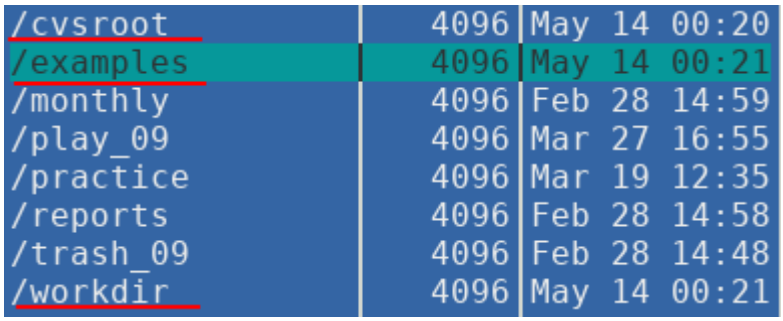
Цель работы

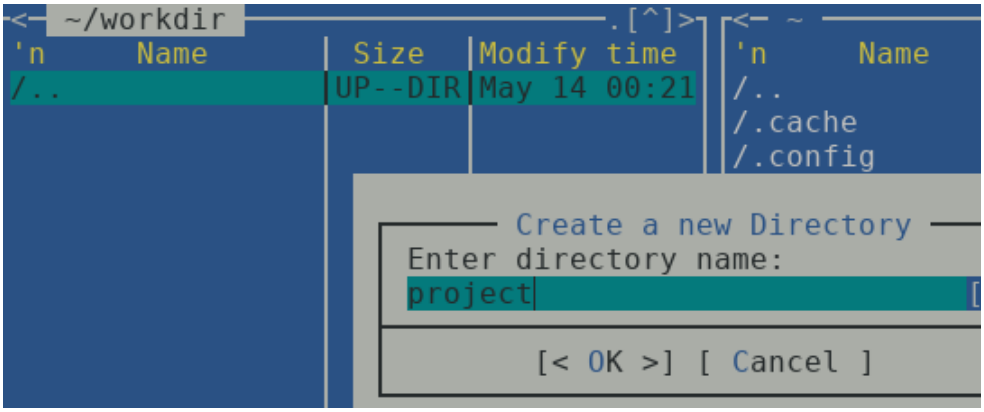
Изучение основных этапов разработки и отладки приложений в ОС Linux, а также приобретение практических навыков по использованию инструментальных средств фонда свободного программного обеспечения при компиляции исходного кода, сборке, отладке и тестировании программ, написанных на языке Си.

1. Запустите файловый менеджер **mc** .

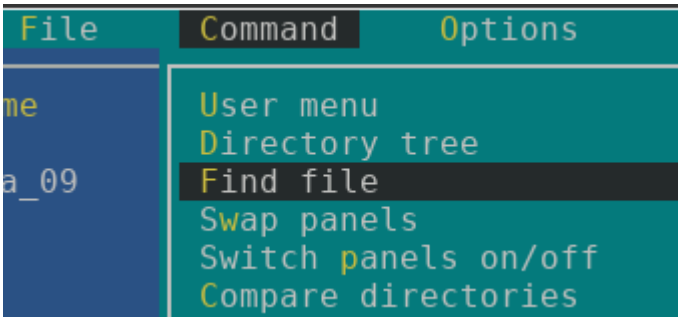


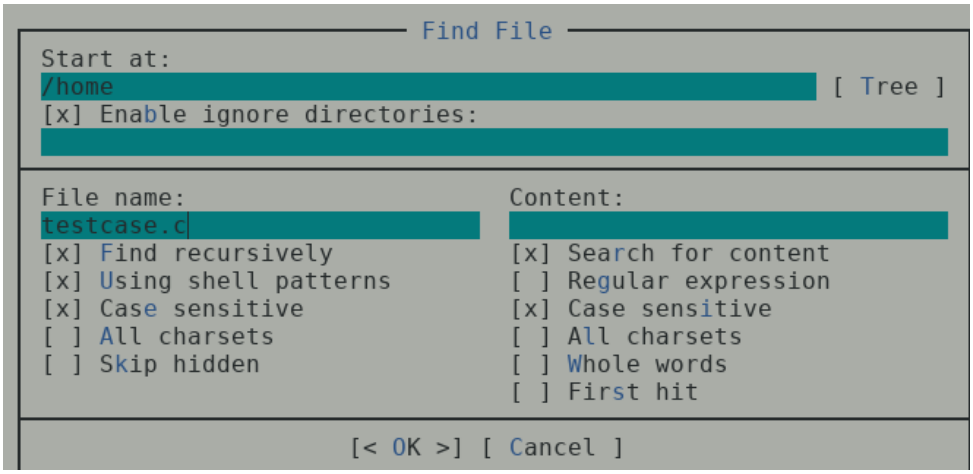
2. В домашнем каталоге создайте каталоги **cvsroot**, **workdir** и **examples**, в каталоге **workdir** создайте подкаталог **project**

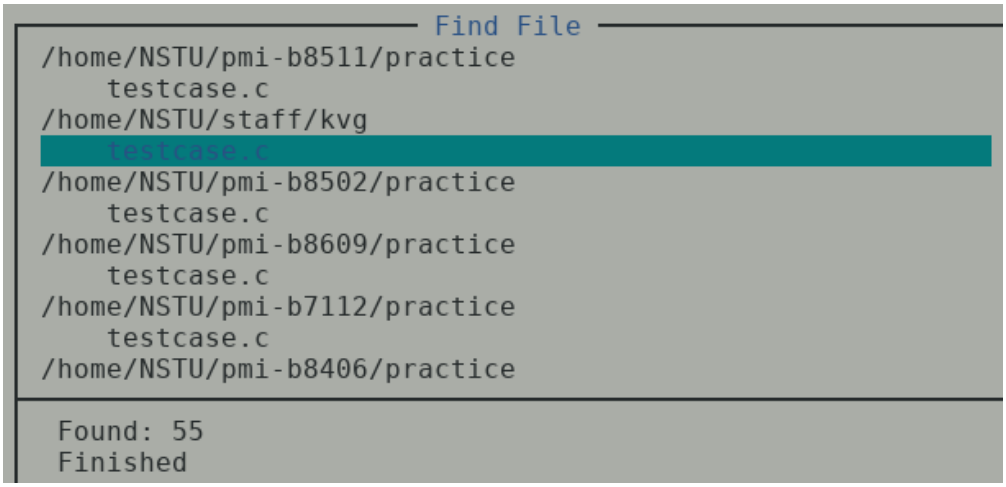




3. Выполните поиск во внешней памяти файла **testcase.c**



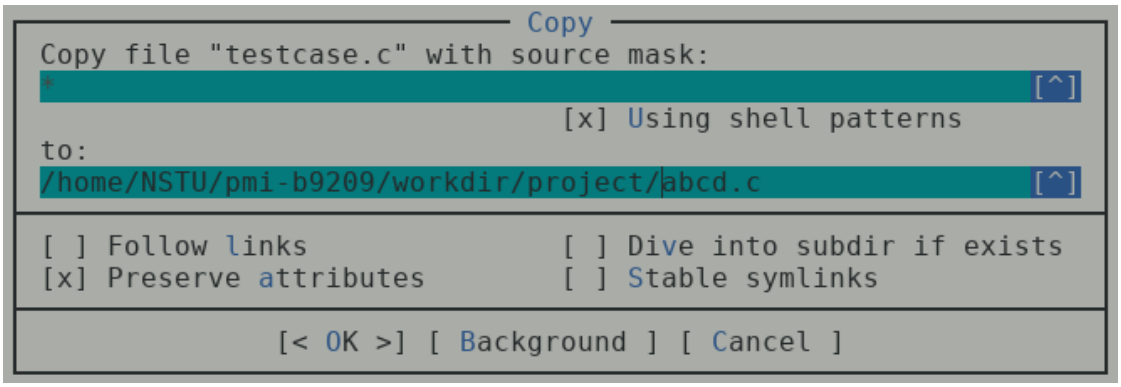




Нужный файл находится в каталоге practice.

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

Выделим файл testcase.c, воспользуемся функцией [5] Copy:



Содержимое testcase.c:

```
1     #include <stdio.h>
2     #include <ctype.h>
3     #include <string.h>
4     /* Manifests for state machine to parse input line. */
5     #define WORD       0
6     #define IGNORE   1
7     /* Globals, used by both subroutines. */
8     char     *Words[BUFSIZ/2];                 /* Worst case, single letters. */
9     int      WordCount;
10    /* Walk through the array of works, find those with the
11     * matching charakter, printing them on stdout. Note that
12     * the NULL charakter will match all words. */
13
14    void PrintWords(wc, match)
15
16    int wc;                          /* Number of words in Words[] */
17    char match;                      /* Attempt to match this charakter. */
18    {  register int    ix;           /* Index in Words[]. */
19       register char   *cp;          /* Pointer for searching. */
20       for (ix=0; ix < wc; ix++) {
21                  cp = Words[ix];
22                  /* Try to match the given character.
23                   * Scan the word, attempting to match,
24                   * or until the end of the word is found. */
25                  while ((*cp) && (*cp++ != match));
26                  if (*cp == match)  /* Found a match? Write the word on stdout. */
27                          (void) printf("%s0, Words[ix]); } return; }
28
29    /* Find words in the gives buffer. The Words[] array is set
30     * to point at words in the buffer, and the buffer modifeid
31     * with NULL characters to delimit the words. */
32    int GetWords (buf)
33    char     buf[];                      /* The input buffer. */
34    {        register char    *cp;       /* Pointer for scanning. */
35             int      end = strlen(buf); /* length of the buffer. */
36             register int  wc = 0;       /* Number of words found. */
37             int      state = IGNORE;    /* Current state. */
```

```
38              /* For each character in the buffer. */
39              for (cp = &buf[0]; cp < &buf[end]; cp++) {
40                  /* A simple state machine to process
41                   * the current character in the buffer. */
42                  switch(state) {
43                  case IGNORE:
44                       if (!isspace(*cp)) {
45                           Words[wc++] = cp;  /* Just started a word? Save it. */
46                           state = WORD;      /* Reset the state. */ } break;
47                  case WORD:
48                       if (isspace(*cp)) {
49                           *cp = '\0';        /* Just completed aword? terminate it. */
50                           state = IGNORE;    /* Reset the state. */ } break; }}
51               return wc; /* Return the word count. */ }
52
53      int  main(argc, argv)  int argc; char *argv[]; { char buf[BUFSIZ], match;
54      /* Check command line arguments. */
55      if (argc < 2) match = ' ';
56      /* No command line argument, match all words. */
57      else match = *++argv[1];   /* match the char after the first - */
58              /* Until no more input on stdin. */
59              while(gets(buf) != (char *)NULL) {
60                   WordCount = GetWords(buf);  /* Paste the input buffer. */
61                   PrintWords(WordCount, match); /* Print the matching words. */ }
62              return(0);   /* Return success to the shell. *
```

5. С помощью редактора vi создайте в каталоге **project** make-файл согласно п. 2.3

```
[pmi-b9209@students ~]$ cd workdir
[pmi-b9209@students workdir]$ cd project
[pmi-b9209@students project]$ vim Makefile
```

```
# Makefile for abcd.c
# Compile abcd.c normaly
 abcd: abcd.c
        gcc -o abcd abcd.c
# Compile abcd.c with debugging
 testabcd: abcd.c
        gcc -o testabcd -g abcd.c
# End Makefile
```

6. Выполните компиляцию программы **abcd.c**. с помощью make-файла, используя правило **abcd**. При необходимости исправьте синтаксические ошибки с помощью редактора **vim**, информацию по ошибкам и их устранению занесите в отчет (номер строки, значение строки до устранения и после устранения ошибки, пояснения).

```
[pmi-b9209@students project]$ cc -o abcd abcd.c
abcd.c: In function ‘PrintWords’:
abcd.c:27:41: warning: missing terminating " character [enabled by default]
                          (void) printf("%s0, Words[ix]); } return; }
                                                         ^
abcd.c:27:27: error: missing terminating " character
                          (void) printf("%s0, Words[ix]); } return; }
                                 ^
abcd.c:32:5: error: expected expression before ‘int’
     int GetWords (buf)
     ^
abcd.c:62:27: error: unterminated comment
              return(0);   /* Return success to the shell. *
                           ^
abcd.c:62:14: error: expected declaration or statement at end of input
              return(0);   /* Return success to the shell. *
              ^
abcd.c:62:14: error: expected declaration or statement at end of input
[pmi-b9209@students project]$
```

Строка 27
(void) printf("%s0, Words[ix]); } return; }
Исправлено на
(void) printf("%s0", Words[ix]); } return; }

Строка 62
/* Return success to the shell. *
Исправлено на
/* Return success to the shell. */}

```
[pmi-b9209@students project]$ gcc -o abcd abcd.c
abcd.c: In function ‘main’:
abcd.c:59:14: warning: ‘gets’ is deprecated (declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]
              while(gets(buf) != (char *)NULL) {
              ^
```

7. В каталоге cvsroot создайте репозиторий CVS.

   [pmi-b9209@students cvsroot]$ cvs -d /home/NSTU/pmi-b9209/cvsroot init

8. Передайте каталог **project** и файл **abcd.c** в репозиторий. При выполнении команды **commit** с помощью редактора **vi** введите комментарий, например: “Пользователь pmi-b9209 передал файл abcd.c под управление CVS”.

[pmi-b9209@students cvsroot]$ cvs -d /home/NSTU/pmi-b9209/cvsroot checkout -l .

```
[pmi-b9209@students workdir]$ cvs -d /home/NSTU/pmi-b9209/cvsroot add project
Directory /home/NSTU/pmi-b9209/cvsroot/project added to the repository

[pmi-b9209@students workdir]$ cvs -d /home/NSTU/pmi-b9209/cvsroot add project/abcd.c
cvs add: scheduling file `project/abcd.c' for addition
cvs add: use 'cvs commit' to add this file permanently
```

```
[pmi-b9209@students workdir]$ cvs commit
cvs commit: Examining .
cvs commit: Examining project
RCS file: /home/NSTU/pmi-b9209/cvsroot/project/abcd.c,v
done
Checking in project/abcd.c;
/home/NSTU/pmi-b9209/cvsroot/project/abcd.c,v  <--  abcd.c
initial revision: 1.1
done
```

9. Запустите исполняемый файл **abcd** и поясните результат запуска.

```
[pmi-b9209@students project]$ ./abcd
^C
```

Ничего не произошло, так как программа содержит ошибки.

10. Перекомпилируйте программу с помощью правила **testabcd** make-файла.

```
[pmi-b9209@students project]$ make testabcd
gcc -o testabcd -g abcd.c
abcd.c: In function ‘main’:
abcd.c:59:14: warning: ‘gets’ is deprecated (declared at /usr/include/stdio.h:638)
              while(gets(buf) != (char *)NULL) {
              ^
```

11. С помощью отладчика **gdb** выполните 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 занесите в отчет результаты тестирования программы в двух вариантах запуска – без параметра и с параметром.

```
[pmi-b9209@students project]$ gdb testabcd
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-80.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/NSTU/pmi-b9209/workdir/project/testabcd...done.
```

```
(gdb) break 55
Breakpoint 1 at 0x400796: file abcd.c, line 55.
(gdb) run
Starting program: /home/NSTU/pmi-b9209/workdir/project/testabcd

Breakpoint 1, main (argc=1, argv=0x7fffffffe198) at abcd.c:55
55          if (argc < 2) match = ' ';
Missing separate debuginfos, use: debuginfo-install glibc-2.17-106.el7_2.1.x86_64
(gdb) step
59                  while(gets(buf) != (char *)NULL) {
(gdb) step
this is a test program
60                       WordCount = GetWords(buf);  /* Paste the input buffer. */
(gdb) step
GetWords (buf=0x7fffffffc0a0 "this is a test program") at abcd.c:35
35                  int      end = strlen(buf); /* length of the buffer. */
(gdb)
36                  register int  wc = 0;       /* Number of words found. */
(gdb) display end
1: end = 22
(gdb) step
37                  int      state = IGNORE;    /* Current state. */
1: end = 22
(gdb)
39                  for (cp = &buf[0]; cp < &buf[end]; cp++) {
1: end = 22
(gdb)
42                      switch(state) {
1: end = 22
(gdb) display state
2: state = 1
(gdb) step
44                          if (!isspace(*cp)) {
2: state = 1
1: end = 22
(gdb)
45                              Words[wc++] = cp;  /* Just started a word? Save it. */
2: state = 1
1: end = 22
(gdb)
```

```
46                                  state = WORD;      /* Reset the state. */ } break;
2: state = 1
1: end = 22
(gdb)
39                  for (cp = &buf[0]; cp < &buf[end]; cp++) {
2: state = 0
1: end = 22
(gdb)
42                      switch(state) {
2: state = 0
1: end = 22
(gdb)
48                           if (isspace(*cp)) {
2: state = 0
1: end = 22
(gdb)
39                  for (cp = &buf[0]; cp < &buf[end]; cp++) {
2: state = 0
1: end = 22
(gdb)
42                      switch(state) {
2: state = 0
1: end = 22
(gdb) break 51
Breakpoint 2 at 0x400772: file abcd.c, line 51.
(gdb) continue
Continuing.

Breakpoint 2, GetWords (buf=0x7fffffffc0a0 "this") at abcd.c:51
51                   return wc; /* Return the word count. */ }
2: state = 0
1: end = 22
(gdb) print wc
$1 = 5
(gdb) step
main (argc=1, argv=0x7fffffffe198) at abcd.c:61
61                       PrintWords(WordCount, match); /* Print the matching words. */ }
(gdb)
PrintWords (wc=5, match=32 ' ') at abcd.c:20
20              for (ix=0; ix < wc; ix++) {
```

```
(gdb) print wc
$2 = 5
(gdb) print match
$3 = 32 ' '
(gdb) step
21                      cp = Words[ix];
(gdb)
25                      while ((*cp) && (*cp++ != match));
(gdb)
26                      if (*cp == match)  /* Found a match? Write the word on stdout. */
(gdb)
20              for (ix=0; ix < wc; ix++) {
(gdb)
21                      cp = Words[ix];
(gdb)
25                      while ((*cp) && (*cp++ != match));
(gdb)
26                      if (*cp == match)  /* Found a match? Write the word on stdout. */
(gdb)
20              for (ix=0; ix < wc; ix++) {
(gdb)
21                      cp = Words[ix];
(gdb) print cp
$4 = 0x7fffffffc0a7 ""
(gdb) print *cp
$5 = 0 '\000'
(gdb) step
25                      while ((*cp) && (*cp++ != match));
(gdb) quit
A debugging session is active.

        Inferior 1 [process 44005] will be killed.

Quit anyway? (y or n) y
```

Исправлено:

Строка 55
if (argc < 2) match = ' ';
на
if (argc < 2) match = '\0';
Присваиваем не пробел, а нулевой символ (символ конца строки).

Строка 27
(void) printf("%s0", Words[ix]); } return; }
на
(void) printf("%s\n", Words[ix]); } return; }
Слова должны выводиться в столбик.

Строка 25
while ((*cp) && (*cp++ != match));
на
while ((*cp) && (*cp != match)){cp++;}
Указатель поиска должен находиться за циклом.

Проверка:

```
[pmi-b9209@students project]$ gdb testabcd
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-80.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/NSTU/pmi-b9209/workdir/project/testabcd...(no debugging
(gdb) run
Starting program: /home/NSTU/pmi-b9209/workdir/project/testabcd
this is a test the abcd program
this
is
a
test
the
abcd
program
```

```
[pmi-b9209@students project]$ gdb testabcd
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-80.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/NSTU/pmi-b9209/workdir/project/testabcd...(no debugging
(gdb) run -t
Starting program: /home/NSTU/pmi-b9209/workdir/project/testabcd -t
this is a test the abcd program
this
test
the
```

12. После получения корректных результатов выполнения программы **abcd** с помощью редактора **vim** в начало отлаженной программы введите комментарий: "Программа abcd отлажена с помощью отладчика gdb дд.мм.гг. бригадой группы ПМ-XX в составе: ФИО1, ФИО2..." и сохраните в репозиторий финальную версию программы.
13. Выведите список изменений файла **abcd.c**, выполненных в ходе отладки программы, занесите список в отчет.

[pmi-b9209@students project]$ cvs -d /home/NSTU/pmi-b9209/cvsroot log abcd.c

RCS file: /home/NSTU/pmi-b9209/cvsroot/project/abcd.c,v
Working file: abcd.c
head: 1.5
branch:

```
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 5;          selected revisions: 5
description:
----------------------------
revision 1.5
date: 2021/05/26 12:34:02; author: pmi-b9209; state: Exp; lines: +1 -0
 Финальная версия
----------------------------
revision 1.4
date: 2021/05/26 12:32:32; author: pmi-b9209; state: Exp; lines: +1 -1
 Синтаксическая ошибка
----------------------------
revision 1.3
date: 2021/05/26 12:30:25; author: pmi-b9209; state: Exp; lines: +13 -6
 Цикл
----------------------------
revision 1.2
date: 2021/05/26 12:14:26; author: pmi-b9209; state: Exp; lines: +1 -1
 match
----------------------------
revision 1.1
date: 2021/05/26 12:07:32; author: pmi-b9209; state: Exp;
Пользователь pmi-b9209 передал файл abcd.c под управление CVS.
```

14. 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.

```
[pmi-b9209@students project]$ du -b testabcd
10683   testabcd
[pmi-b9209@students project]$ strip -s testabcd
[pmi-b9209@students project]$ du -b testabcd
6256    testabcd
```

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

```
[pmi-b9209@students project]$ cp abcd.c ../../examples/
```

16. Выполните разбиение полностью отлаженной программы **abcd** на функции в соответствии с номером бригады из таблицы 26. 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.

/* Программа abcd отлажена с помощью отладчика gdb 26.05.21 бригадой группы ПМ-92 в составе: Иванов Владислав, Кутузов Иван. */

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#include "getwords.h"

/* Manifests for state machine to parse input line. */
#define WORD       0
#define IGNORE   1
/* Globals, used by both subroutines. */
char     *Words[BUFSIZ/2];                /* Worst case, single letters. */
int      WordCount;

extern void PrintWords(int wc, char match, char *Words[]);
/* 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. */

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 input on stdin. */
        while(gets(buf) != (char *)NULL) {
            WordCount = GetWords(buf);  /* Paste the input buffer. */
            PrintWords(WordCount, match); /* Print the matching words. */ }
        return(0);   /* Return success to the shell. */}
```

Для Лены:

Сивак просила вписать в отчёт также и код модулей (getwords и printwords). Кобылянский, скорее всего, тоже попросит.

17. Выполните сборку программы в соответствии вариантом задания, используя неявный вызов компоновщика и задав имя исполняемого файла **adcd2_1**; проверьте корректность работы программы и занесите в отчет результаты ее тестирования.

```
[pmi-b9209@students project]$ gcc -o abcd2_1 abcd2.c
[pmi-b9209@students project]$ ./a.out
test
test
```

18. Выполните поиск во внешней памяти сервера каталога **obj**, скопируйте из него все файлы в каталог **example**, поясните в отчете назначение скопированных файлов.

```
[pmi-b9209@students project]$ cp -r /home/NSTU/staff/kvg/obj obj/
[pmi-b9209@students project]$ ls obj
crt1.o  crtbegin.o  crtend.o  crti.o  crtn.o
```

19. Выполните сборку программы в соответствии вариантом задания, используя явный вызов компоновщика. Результатом сборки должны быть исполняемый файл **adcd2_2** и карта памяти **adcd2_map**; проверьте корректность работы программы и занесите в отчет результаты ее тестирования.

```
[pmi-b9209@students project]$ gcc -c abcd2.c
[pmi-b9209@students project]$ ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o abcd2_2 -Map= abcd2_map obj/crt1.o obj/crti.o obj/crtbegin.o abcd2.o -lc obj/ crtend.o obj/crtn.o
[pmi-b9209@students project]$ ./abcd2_2
this is a test the abcd program
this
is
a
test
the
abcd
program
[pmi-b9209@students project]$ ./abcd2_2 -t
this is a test the abcd program
this
test
the
```

20. Из карты памяти **adcd2_map** определите размеры машинного кода модулей **abcd2.o**, **printwords.o** и **getwords.o**, сравните их с размерами исходного и объектного кода этих модулей (файлы типа .c и .o). Результат представьте в виде таблицы 27, все данные должны быть подтверждены скриншотами.

Имя модуля (функции)	Исходный, байт	Объектный, байт	Машинный код, байт
printwords			
getwords		–	–
abcd2			

Для Лены:

Если у тебя файл .h (у меня это getwords), то в двух последних столбцах ничего.

Как заполнять таблицу: сделать все шаги выше, появится файл abcd2_map. Его открыть вимчиком, а потом спросить Влада, откуда что брать.

В отчёт также нужно скриншот добавить этого файла, показать откуда что брали.

21. Добавьте в make-файл, разработанный при выполнении п.5, два новых правила, реализующие п. 17 и 19 задания. Проверьте корректность его работы.

...

abcd2_1: abcd2.c

gcc -o abcd2_1 abcd2.c

abcd2_2: abcd2.c

gcc -c abcd2.c

ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o abcd2_2 -Map=

abcd2_map obj/crt1.o obj/crti.o obj/crtbegin.o abcd2.o -lc obj/crtend.o obj/crtn.o

…

```
[pmi-b9209@students project]$ make abcd2_1
make: `abcd2_1' is up to date.
[pmi-b9209@students project]$ make abcd2_2
make: `abcd2_2' is up to date.
```