Лабораторное занятие 2. Установка пакетов-портов в ОС FreeBSD

Задание 1. Настройка сети

1. Установка имени компьютера.

Ваша виртуальная машина (ВМ) с установленной ОС FreeBSD может работать в компьютерной сети. Для этого необходимо чтобы ВМ имела имя, которое задается с помощью системной переменной `hostname`. Возможно, `hostname` у вас уже настроена, тогда сейчас ничего делать не нужно.

Для проверки имени ВМ воспользуемся командой hostname: она должна что-нибудь вывести справа от приглашения ко вводу

```
asdf1#
```

Если Вы ничего не увидели, имя ВМ можно установить двумя способами.

А. Исправить файл `/etc/defaults/rc.conf`, установив в нем параметр равным придуманному Вами имени

```
hostname=""                    # Set this!
```

Б. Поскольку первый способ может потребовать знаний по изменению параметров файлов, можно сделать это проще. Переопределения для этого файла задаются в `/etc/rc.conf`,
который изначально должен быть пустым. Добавить в него строку можно выполнив команду

```
echo 'hostname="asdf1"' >>/etc/rc.conf
```

Здесь «asdf1» - пришедшее на ум имя.

2. Настройка сети через sysinstall

А. Запустите утилиту sysinstall, в меню Configure выберите пункт Networking – рис. 1.

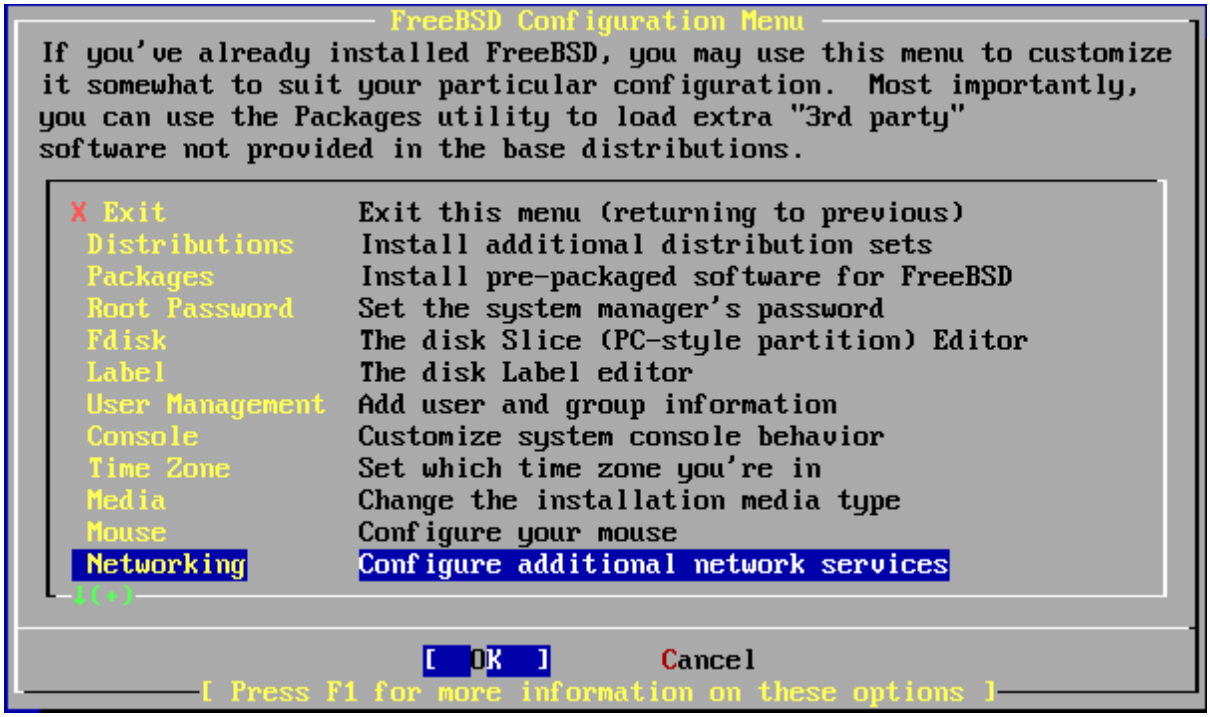

Рис. 1.

Б. Выберите пункт Interfaces, и настройте интерфейс de0 (для виртуальной машины vmware – **`lnc0`**).

В. Откажитесь от Ipv6 (новая и расширенная версия протокола IP), и согласитесь с DHCP (протокол автоматической настройки сети с сервера). Вы увидите диалоговое окно (ДО), показанное на рис. 2. Перемещайтесь с помощью клавиши <Tab> до кнопки <OK> и нажмите на клавишу <Enter>.

Все эти параметры можно было задать вручную, но в сетях с сервером проще использовать DHCP.

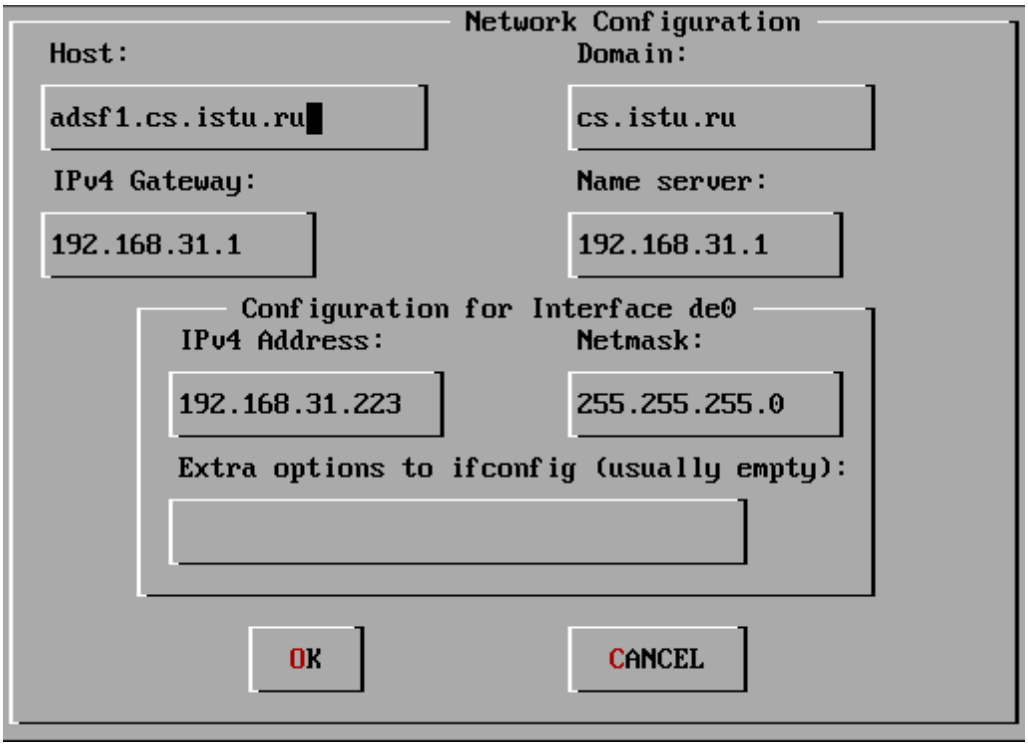

Рис. 2.

DHCP (англ. *Dynamic Host Configuration Protocol* — протокол динамической конфигурации узла) — это сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Для этого компьютер обращается к специальному серверу, называемому *сервером DHCP*. Сетевой администратор может задать диапазон адресов, распределяемых среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве крупных сетей TCP/IP.

3. Проверка сети.

```
adsf1# ping cs
PING root.cs.istu.ru (192.168.31.1): 56 data bytes
64 bytes from 192.168.31.1: icmp_seq=0 ttl=64 time=4.668 ms
64 bytes from 192.168.31.1: icmp_seq=1 ttl=64 time=0.966 ms
64 bytes from 192.168.31.1: icmp_seq=2 ttl=64 time=1.227 ms
64 bytes from 192.168.31.1: icmp_seq=3 ttl=64 time=0.836 ms
64 bytes from 192.168.31.1: icmp_seq=4 ttl=64 time=1.450 ms
64 bytes from 192.168.31.1: icmp_seq=5 ttl=64 time=1.264 ms
64 bytes from 192.168.31.1: icmp_seq=6 ttl=64 time=1.698 ms
64 bytes from 192.168.31.1: icmp_seq=7 ttl=64 time=0.845 ms
64 bytes from 192.168.31.1: icmp_seq=8 ttl=64 time=0.968 ms
64 bytes from 192.168.31.1: icmp_seq=9 ttl=64 time=1.487 ms
```

Рис. 3.

А. Наберите команду `ifconfig`. Должен появиться список интерфейсов, среди которых должен быть de0.

Б. Наберите `ping cs`. Вы увидите примерно следующее – рис. 3. Это значит, что сеть работает правильно.

Задание 2. Запуск серверов ftp (пересылка файлов) и ssh (удаленный вход в систему)

1. Вернитесь в Configure/Networking (как в задании 1) и включите пункты ssh и inetd – см. рис. 4.

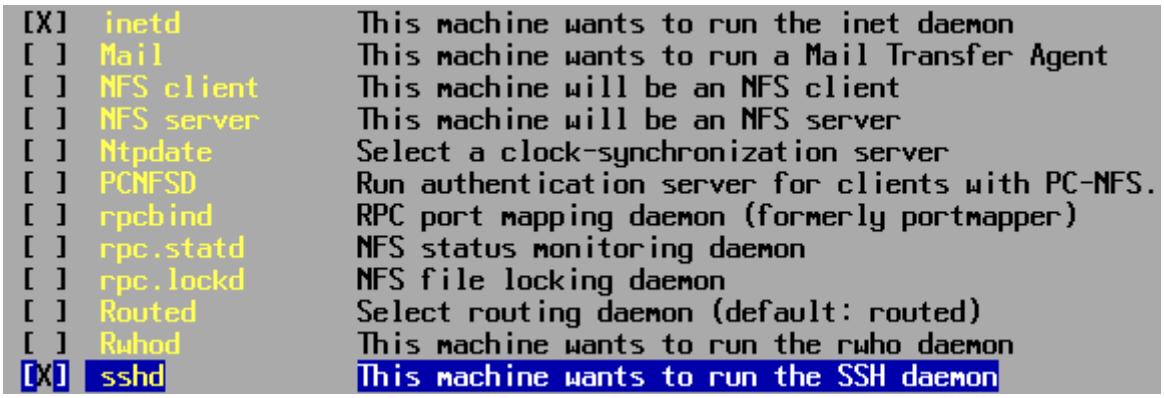

Рис. 4.

При включении `inetd` вам предложат включить несколько сервисов - сделайте это. Далее автоматически будет запущен редактор `vi` с загруженным в него настроечным файлом `/etc/inetd.conf`, который содержит настройки для сетевого демона `inetd`. Вам нужно раскомментировать строчку `ftp` – стереть символ '#'

```
ftp     stream  tcp     nowait  root    /usr/libexec/ftpd       ftpd -l
```

Закройте редактор `vi` с сохранением информации (команда ":wq") и перезапустите систему.

2. В процессе загрузки вам будет предложено понажимать клавиши, чтобы накопить случайные данные для генерации ключей ssh. В данном случае это не важно, однако на реальных системах это определяет безопасность.

Теперь, зная свой `ip`-адрес (ifconfig), можно работать при помощи утилиты `far` по протоколу `ftp` – этой возможностью мы пока пользоваться не будем.

Задание 3. Работа с протоколом `ftp` через встроенного клиента

```
adsf1# ftp
ftp> open cs
Connected to root.cs.istu.ru.
220 Welcome to cs.istu.ru
Name (cs:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||56325|)
150 Here comes the directory listing.
drwxr-xr-x   13 1000     1001         4096 May 28  2007 confer
drwxr-xr-x    6 1000     1001         4096 Dec 28 11:42 distrib
drwxrwxr-x   12 1000     1001         4096 Feb 21 11:43 docs
drwx------    2 0        0           16384 Sep 08  2006 lost+found
drwxr-xr-x    4 1000     1001         4096 Dec 27 11:58 utils
-rw-r--r--    1 0        0              57 May 10  2006 welcome.msg
226 Directory send OK.
ftp>
```

Рис. 5.

`ftp` - это простой протокол обмена файлами между клиентом и сервером. Самый простой клиент для него – ftp, встроенный в любую операционную систему. Работа с ним очень проста – пример приведен на рис. 5. *Вам нужно его повторить*.

Чтобы зайти на сервер кафедры, я запустил клиент, набрал команду open cs (можно было использовать `ip`-адрес). Имя пользователя `anonymous` используется тогда, когда надо предоставить сервис всем желающим. Иногда может потребоваться ввести корректный адрес электронной почты в качестве пароля. Для работы с каталогами применяются команды ls и cd. Скачать какой-то файл можно командой get, при этом он скачается в текущую директорию на локальном компьютере – в нашем случае это директория на ВМ. Список команд можно получить командой help, справку о назначении команды – `help <команда>`. Для завершения работы с клиентом используем команду exit.

Задание 4. Установка файлового менеджера `mc`

1. Установка коллекции портов (опционально).
   - Попробуйте сделать `cd` в `/usr/ports`.
   - Если это вызовет ошибку, необходимо установить коллекцию портов. Если нет, переходите к п. 2.
   - Запустите `sysinstall`.
   - Перейдите в `Configure/Distributions`.
   - Выделите пункт `ports` и запустите установку.
2. Теперь должна существовать директория `/usr/ports/distfiles`. Нужно перейти в нее и, используя ftp, скачать с ftp://cs/distrib/OS/freeBSD/lab2 все файлы, сохраняя структуру директорий.
   - Сначала надо выполнить команду `cd /usr/ports/distfiles`.
   - Затем - `mkdir gnome2`.
   - Затем надо войти на `ftp`-сервер кафедры и выполнить команду `cd`

     ```
     ftp> cd distrib/OS/freeBSD/lab2/
     ```

   - После этого можно выполнить команды `mget gnome2` и `mget *`.

На этом этапе содержимое директории `/usr/ports/distfiles` нашей ВМ должно выглядеть так.

```
[root@adsf1 /usr/ports/distfiles]# ls -l
total 6738
drwxr-xr-x  2 root  wheel      512 Mar 11 20:35 gnome2
-rw-r--r--  1 root  wheel  2921483 Dec 18  2005 libtool-1.5.22.tar.gz
-rw-r--r--  1 root  wheel  3928370 Jul 23  2005 mc-4.6.1.tar.gz
```

3. Перейти в `/usr/ports/misc/mc` и набрать `make && make install`.
   Установка закончится примерно так, как показано на рис. 6. Сменив пользователя (выполнить команду `exit` и снова войти в систему) и набрав команду `mc`, мы увидим знакомые голубые окна файлового менеджера – рис. 7.

```
gmake[2]: Nothing to be done for `install-exec-am'.
gmake[2]: Nothing to be done for `install-data-am'.
gmake[2]: Leaving directory `/usr/ports/misc/mc/work/mc-4.6.1'
gmake[1]: Leaving directory `/usr/ports/misc/mc/work/mc-4.6.1'
===>   Compressing manual pages for mc-4.6.1_3
===>   Registering installation for mc-4.6.1_3
===> SECURITY REPORT:
      This port has installed the following files which may act as network
      servers and may therefore pose a remote security risk to the system.
/usr/local/bin/mc

      If there are vulnerabilities in these programs there may be a security
      risk to the system. FreeBSD makes no guarantee about the security of
      ports included in the Ports Collection. Please type 'make deinstall'
      to deinstall the port if this is a concern.

      For more information, and contact details about the security
      status of this software, see the following webpage:
http://www.ibiblio.org/mc/
```

Рис. 6.

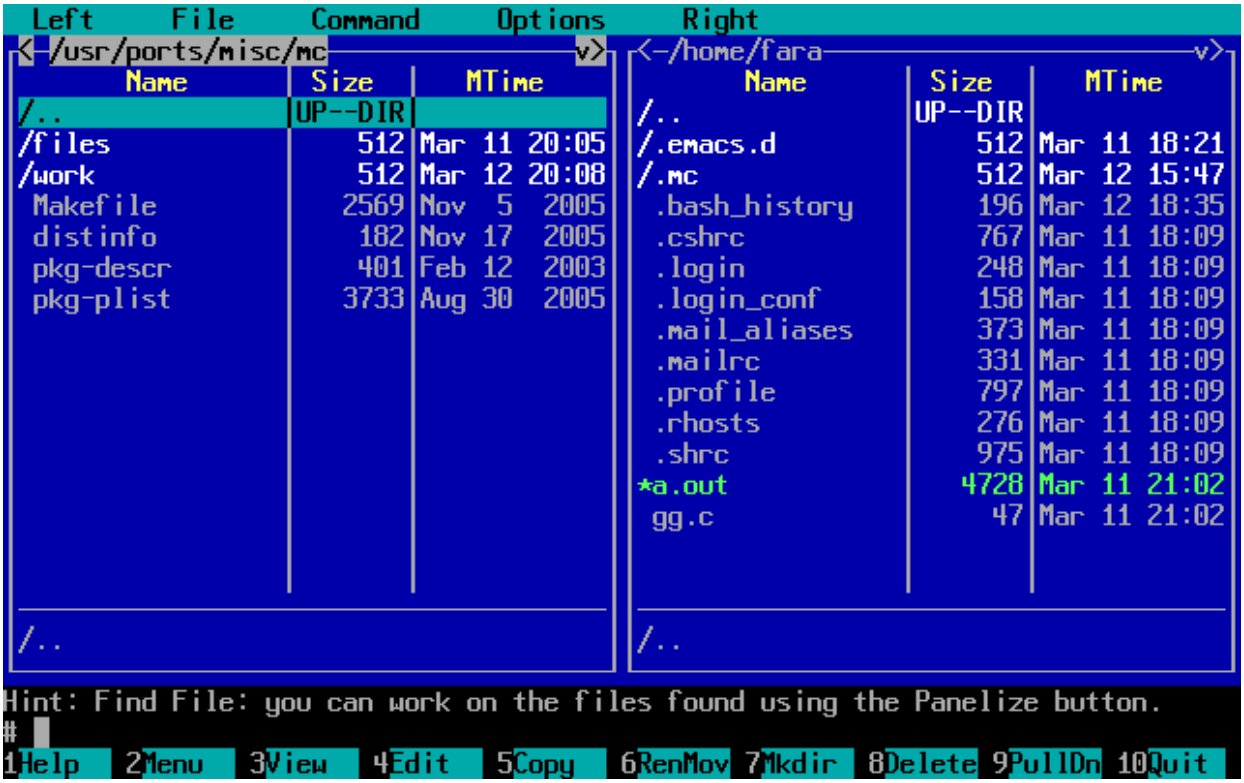

Рис. 7.

Задание 5. Создание стандартного `make`-файла для своей программы

Проектирование и написание `make`-файлов – один из важных элементов в работе программиста под управлением ОС типа Unix. В изучаемой ОС FreeBSD с помощью `make`-файлов производится установка дополнительного программного обеспечения, поставляемого на уровне исходных кодов – так называемые порты (ports). Этим способом мы воспользовались при установке программы файлового менеджера mc. Теперь займемся созданием упрощенного порта.

1. Написать простую программу с использованием библиотеки `ncurses`.

Библиотека `ncurses` служит для создания программ, работающих с окнами, управления курсором в текстовом режиме, задания цвета символов и фона и т.д. Для демонстрации возможностей библиотеки в рассмотренном ниже примере в середину пустого экрана выводится сообщение `"Hello, World!"`. После нажатия на любую клавишу оно исчезает, и программа заканчивает работу. Моя программа выглядит так.

```
#include <curses.h>
#include <stdio.h>
#include <string.h>
int main()
{
          char s[]="Hello, World!";
          int ls=strlen(s);
          initscr(); cbreak(); noecho();
          mvprintw( LINES/2, COLS/2-ls/2, "%s", s );
          getch();
          echo();
          nocbreak();
          endwin();
          return 0;
}
```

Для справки. Функции из библиотеки `ncurses`

`nocbreak()/cbreak()` -- включение/выключение буфферизации ввода.

`nodelay(stdscr, bool Value)` – включение/выключение режима блокировки `getch`.

При включении режима `nodelay` `getch()` вместо того чтобы ожидать ввода символа вернет `ERR`.

`move(int y, int x)` – перемещает курсор в позицию `(x,y)`

`printw()` -- аналог `printf` для `ncurses`. Есть версия `mvprintw` которая первыми двумя аргументами берет `y` и `x` и перемещает курсор туда перед выводом.

`int usleep(useconds_t microseconds)` – подождать определенное время (в микросекундах).

`int getch()` -- Функция читает символ со стандартного ввода. В случае если включена буферизация ввода, символ будет получен только после нажатия пользователем `enter`, иначе символ будет получен сразу.

В случае если включено эхо `(echo() )`, символ будет выведен на экран, иначе нет.

В случае если включен режим `nodelay` то вызов функции `getch()` не будет дожидаться ввода символа. Вместо этого в случае отсутствия символа в буфере он вернет значение ERR.

Ниже приведен пример программы, которая каждые полсекунды опрашивает клавиатурный буфер. Если пользователь нажимал на клавиши, то печатается сообщение **"Key was pressed".** Если нажатий в последние полсекунды не было, то печатается **"No key was pressed!"**. Получите у преподавателя задание на составление подобной «экранной» программы.

```
#include <stdio.h>
#include <curses.h>

int main(void)
{ int c=32, c1, x;
initscr();
scrollok(stdscr, true);      //включить прокрутку экрана
nodelay(stdscr,true);        //включить неблокирующий режим для getch()
nocbreak();                  //выключить буфферизацию
while (c != 27)
{ if ( (x=getch())!=ERR )              //if keypressed then readkey...
  {  addstr(" Key was pressed\n");     //addstr – самая простая функция вывода. //
                                       выводит одну строку.
    do   if ( x == 27 ) c=(char)x;
    while  ((x=getch())!=ERR);
  }
  else addstr("No key was pressed!\n");
  usleep(500000); /0.5 секунды!
} // while
nodelay(stdscr,false);       //вернуть getch() на место
return 0;
}
```

Рис. 8. Пример «экранной» программы с опросом клавиатурного буфера

2. Стандартные цели для `makefile`.

Для того чтобы быть удобным множеству пользователей, ваш `makefile` должен поддерживать некоторые стандартные цели.

`all` – скомпилировать программу. Должна быть первой целью в файле, чтобы при простом наборе `make` выполнялась именно компиляция.
`clean` – очистка директории сборки от бинарных файлов.
`install` – копирование бинарных файлов программы в пути по умолчанию.
`uninstall/deinstall` – удаление всех созданных нами с помощью цели `all` бинарных файлов из системы.

3. Написание цели `all`.

Рассмотрим подробно процесс компиляции. Сначала из каждого с файла создается объектный файл (с расширением `.o`). Затем из одного или нескольких о файлов собирается исполняемый файл. Для того чтобы программа могла использовать библиотеку `ncurses`, надо написать `-lncurses`, чтобы линкер слинковал с ней.

```
all: compile

compile: chello

chello: curse.o
        cc curse.o -lncurses -o chello
```

Для получения объектного файла из исходного добавляется следующая цель

```
curse.o: curse.c
         cc -c curse.c
```

В этом примере файлы с исходным текстом и объектным кодом называются `curse`, а исполняемый - `chello`.

4. Написание цели `clean`.

Цель должна просто удалить исполняемый файл и все файлы типа .o.

```
clean:
        rm *.o chello
```

5. Написание цели `install` и `uninstall/deinstall`.

Эта цель должна скопировать исполняемый файл в `/usr/local/bin` (`/usr` означает, что программа не системная, `local` – что собрана на этой системе, а не поставлена пакетом, `bin` – что это исполняемый файл).

```
install: chello
        cp chello /usr/local/bin/chello

deinstall uninstall:
        rm /usr/local/bin/chello
```

Итоговый `makefile` выглядит так:

```
all: compile

compile: chello

chello: curse.o
        cc curse.o -lncurses -o chello

curse.o: curse.c
        cc -c curse.c

clean:
        rm *.o chello

install: chello
        cp chello /usr/local/bin/chello

deinstall uninstall:
        rm /usr/local/bin/chello
```

Выполнять цели можно так:

`make all install`	– для компиляции и установки программы в нужную директорию;
`make clean`	– для удаления объектных и исполняемых файлов.

Продемонстрируйте возможности Вашего `makefile`.

Много полезных и удобных приемов можно узнать в статье:
`http://www.linux.org.ru/books/GNU/Gmake.htm`

Задание 6. Использование отладчика `gdb` для отладки программ

Предположим, написана программа `bug.c`, например такая:

```
#include <stdio.h>
int main()
{
          int i;
          for(i=10;i>=0;--i)
          {
                    printf("%d ",10/i);
          }
          puts("\n");
          return 0;
}
```

При запуске программа выдает ошибку: Floating point exception.

Для отладки при помощи `gdb` надо проделать следующие действия:

1. Собрать программу с ключом отладки `-g`: `gcc -g bug.c -o bug`
2. Запустить программу под отладчиком: `gdb bug`
3. Появится приглашение отладчика ко вводу, в ответ на которое можно будет вводить команды – см. рис. 8.
   Для того чтобы просмотреть программу вокруг места выполнения, используется комада `list`; ей можно дать номер строки. Если просто нажать `<Enter>`, то будет повторена последняя команда. Отладчик `gdb` распознает команды даже по первой букве.

```
  (gdb) l 1
1       #include <stdio.h>
2
3       int main()
4       {
5               int i;
6               for(i=10;i>=0;--i)
7               {
8                       printf("%d ",10/i);
9               }
10              puts("\n");
(gdb)
11              return 0;
12      }
```

Рис. 8. Пример вывода отладчика

Следующий шаг – установить точку останова. Для этого служит команда `break` с аргументом – номером строки.

```
(gdb) b 6
Breakpoint 1 at 0x80483a5: file bug.c, line 6.
```

Теперь можно и запустить программу. Она дойдет до строки с точкой останова и снова выведет приглашение. Для запуска служит команда `run`.
Чтобы сделать шаг по программе, используется команда `next`.
В случае, когда следующий оператор содержит вызов функции и нам необходимо войти внутрь этой функции с целью изучения ее поведения, используется команда `step`.
Чтобы вывести содержимое переменной, надо использовать `print <имя>`.

```
(gdb) p i
$4 = 7
```

Просматривать по шагам циклы – скучное занятие. Чтобы запустить программу на дальнейшее выполнение, нужно ввести `continue` или `c`.

```
  Program received signal SIGFPE, Arithmetic exception.
0x080483b4 in main () at bug.c:8
8                       printf("%d ",10/i);
```

Ошибка почти найдена! Если посмотреть на содержимое `i`, то мы увидим 0, на который мы и пытаемся поделить (!).