Лабораторная работа №8 по курсам «Основы информатики» и «Программно-аппаратные средства информатики» на 2011/12 уч. год.

Системы программирования на языке Си

UNIX-серверы терминального класса оснащены многоязычными системами программирования (СП). На платформе DEC Alpha это GNU Compiler Collection 4.1.3 в среде NetBSD 5.1.0. На платформе Intel установлен GCC версии 4.2.2. Также имеются такие впечатляющие средства, как p2c СП GNU, mono C# 2.10.2 и F#, Python, Perl, Haskell, Common Lisp, Scheme, Ruby, PHP, Javascript, Java (OpenJDK), Google Go, Erlang, BrainFuck и Whitespace. Язык C♪ (Си-бемоль) тоже присутствует!

Помимо компиляторов, в состав системы программирования Си обычно входят препроцессоры (отдельный – **cpp** – или интегрированный), отладчики (**gdb**), верификаторы (**lint**), профилёры (**gprof**), форматоры (**indent**), редактор связей (**ld**), ассемблер (**as**), библиотеки (в том числе: математическая, языковой среды и системных вызовов ОС UNIX), документация, заголовочные файлы и многое другое, в том числе и сам язык программирования – входной язык СП, который в конкретных случаях может иметь особенности. Язык Си хорошо стандартизован, имеются два стандарта (ISO 9899-90 и ISO 9899-99), определяющие и библиотеку языка.

Подробнее о системах программирования GNU можно прочитать в соответствующих документах практикума, в том числе и распространяемых на CD ROM, а также в электронных мануалах ОС UNIX (**man** и **info** для команд **cc** или **gcc**). Для СП GNU (GCC) имеется обстоятельная книга [1].

Для подготовки и редактирования исходных текстов программ рекомендуется использовать развитый текстовый редактор Emacs, настроенный на синтаксис нескольких языков. Ещё один популярный среди программистов редактор — Vim. Тексты программ могут быть также подготовлены заранее в домашних условиях в среде MS Windows в любом *текстовом* редакторе (jEdit, Notepad++, MultiEdit и т. п.) как *текстовые файлы стандарта UNIX* с передачей в систему с помощью **ftp** с необходимой перекодировкой и преобразованием формата (**iconv**, **dos2unix**).

Цель лабораторной работы — изучение конкретной системы программирования на Си и получение навыков подготовки текстов и отладки программ. В рамках работы необходимо:

- изучить и освоить возможности лабораторной СП по содержащейся в практикуме документации и другим источникам, в том числе основные этапы процесса компиляции и подготовки программ к выполнению (см. программу зачёта №4).
- составить и отладить простейшую программу на Си в *терминальном классе*.
- изучить различные системы программирования на Си на других платформах: GNU, MS Visual Studio и др. на *домашних компьютерах*.

Система программирования GNU ориентирована прежде всего на использование на многих вычислительных системах (платформах) и в её состав не входит интегрированная среда разработчика (IDE). Редактирование исходных текстов программ и управление процессом программирования и отладки осуществляется программистом в каждой конкретной операционной среде по-своему. Существует несколько универсальных оболочек СП GNU, пригодных для этой цели, например rhide, CodeWarrior, gide и др. Однако, поскольку система программирования изучается нами изнутри, с профессиональной точки зрения, то целесообразно вручную манипулировать компонентами СП. В качестве типовой оболочки СП также могут использоваться **emacs** или **vim**, позволяющие прямо из редактора текстов осуществлять запуск компилятора, управление процессом отладки, включая навигацию по ошибочным строкам исходной программы,.

Основные этапы создания программ:

1. **Запуск редактора текстов** для редактирования или первоначального набора программы:

   emacs lab8.c либо **emacs +58 lab8.c** (для перехода к строке 58)

После редактирования текста программы рекомендуется Emacs перевести в фоновый режим `Ctrl+Z`, возобновляя при необходимости командой **fg**.

2. **Компиляция программы** (в ключах компилятора обязательно задать проверку соблюдения стандарта языка!) и создание (в случае отсутствия синтаксических ошибок) выполнимой программы (`a.out`):

   cc lab8.c либо **gcc -std=c99 -Wall -pedantic lab8.c**

Эти действия могут быть автоматизированы с помощью знаменитой утилиты **make**, изучаемой во II семестре.

Ранее команда **cc** использовалась для вызова «родного» компилятора Си данной версии UNIX. Сегодня **cc**, как правило, это **gcc**.

3. При обнаружении синтаксических ошибок компилятор выдаёт сообщения, и необходимо вернуться к п. 1 (возобновив emacs!) для их исправления. Исправления удобно вести, пользуясь списком номеров строк с ошибками, выданным компилятором. Если ошибок много, то начинать исправления надо с первой ошибки, так как многие последующие ошибки часто являются следствием неправильного восстановления состояния компилятора после первых ошибок (т. н. *наведённые* ошибки). Если список ошибок не помещается в окно терминала, то его можно рассматривать по частям, прокручивая вывод терминала или перенаправляя вывод компилятора (поток ошибок) на пейджер `more` (или `less`):

```
gcc -std=c99 -Wall -pedantic lab8.c 2>&1 | more
```

4. В случае, если ошибок не было, **запуск исполняемого файла** a.out из *текущего* (именно!) каталога для тестирования программы:

`./a.out` либо **`gcc -std=c99 -Wall -pedantic lab8.c && ./a.out`** (полуавтоматическая компиляция перед запуском).

*Явное указание текущей директории (**`./`**) позволяет избежать подставок чужих программ с именем a.out из предписанных пользователю путей поиска выполняемых программ, запускаемых от его имени и с его полномочиями!*

5. В случае обнаружения семантических (смысловых) ошибок, необходимо проанализировать текст программы с целью их обнаружения. Если не удаётся найти ошибки по тексту программы, следует воспользоваться приёмами отладки с помощью ЭВМ.

6. После исправления ошибок в исходном тексте программу *следует **заново** откомпилировать* (п. 2).

В данном примере `lab8.c` — имя файла с исходным текстом программы, которое может выбираться произвольно в соответствии, например, с назначением программы (например, `proba.c`). Суффикс **`.c`** указывает на то, что файл содержит исходный текст программы на Си.

Для *отладки программы* (поиска и устранения ошибок, выявленных при тестировании) может использоваться соответствующая компонента СП — отладчик (которому посвящена отдельная *лабораторная работа №10*) либо средства языка программирования (проверки, промежуточные печати и др., которые впоследствии удаляются из программы или заключаются в комментарии). Не следует пренебрегать отладкой программы «по тексту» (ручная прокрутка с проговариванием вслух и с записью ходов и т. п.), поскольку такие навыки весьма полезны на контрольных работах, коллоквиумах и экзаменах (*особенно на письменных*).

Отладка считается законченной, если программа на некотором достаточно полном наборе тестов дает правильные результаты. Первоначальный набор тестов студент составляет самостоятельно на основании условия задачи и особенностей программы *до* начала программирования и отладки. Тесты заранее включаются в бланк отчёта по лабораторной работе (п. 7) и предъявляется преподавателю.

Прошедшая этот набор тестов программа предъявляется на тестирование преподавателю, который проводит инспекцию её текста и тестирует её как на авторских, так и на своих тестах, возможно, во внеаудиторное время (offline), с коммуникациями по e-mail (при наличии). Программа может быть отклонена преподавателем как по причине неработоспособности, так и ввиду несоответствия заданию или требованиям практикума (ввод-вывод, стиль программирования, используемые нестандартные конструкции языка или нетематические методы решения и др.). Следует ответственно подходить к самостоятельному тестированию, так как сдача программы не с первого раза может привести к снижению оценки. Хорошим средством предварительного тестирования может служить взаимное тестирование программ студентами, так как со стороны легче заметить ошибки и неточности в программе. Научиться тестировать чужие программы тоже весьма полезно.

В случае успешного тестирования преподаватель делает отметку в лабораторном журнале (*кондуите*). Документом, свидетельствующем об окончании тестирования программы, является печатный протокол, выполненный студентом в лабораторных условиях (*именно!*) и подписанный *своим* преподавателем. В случае ведения электронного документооборота ставятся электронные подписи студента и преподавателя, материалы работы депонируются и включаются в реестр.

Процесс протоколирования программ на Си соответствует рассмотренному ранее сценарию с обязательным включением в протокол следующих пунктов:

1. Листинг исходного текста программы (**`cat имя-файла`**, включая **`cat Makefile`**, если он используется).
2. Команда компиляции этой программы (**`cc`** или **`gcc -v`** имя-файла, либо **`make`**). Ключ `-v` в дальнейшем опускают.

3. Распечатка файла тестов, если тестирование ведётся переназначением стандартного входного файла на заранее заготовленный текстовый файл с тестовыми данными.
4. Результаты тестирования полученной программы на достаточно полном наборе тестов (**`./a.out`**, быть может несколько раз, хотя окончательное тестирование (как правило!) должно идти на пакете тестов, считываемых в цикле до конца входного файла).

В ходе лабораторной работы №8 необходимо продемонстрировать преподавателю работу с СП на простых примерах программ печати приветственного сообщения и т. п.:

```
/* Лабораторная работа 8, вариант 13.
 * Студентка гр.08-111 А. Лавлейс */

#include <stdio.h>

int main(){
      printf("Hello, Charles!\n");
      return 0;
}
```

Более подробно с проблемой программирования задач печати приветственных сообщений на всевозможных языках программирования можно ознакомиться, посетив Web-страницы:

http://www.gnu.org/fun/jokes/helloworld.html,
http://en.wikipedia.org/wiki/Hello_world_program_in_esoteric_languages,
http://www2.latech.edu/~acm/HelloWorld.shtml или http://www.ntecs.de/old-hp/uu9r/lang/html/lang.en.html.

С целью освоения системы программирования следует внести в программу синтаксические ошибки (если они не были допущены ранее). Например, сделать опечатку в ключевом слове, не закрыть комментарий или строку литер; внести ошибку времени выполнения (распечатать текущее значение переменной с незаданным начальным значением, попытаться поделить на 0, вызвать переполнение и др.) и изучить соответствующие диагностические сообщения. Для изучения состава СП запуск необходимо производить в т.н. болтливом режиме (с ключом **`-v`**), при котором отображаются основные этапы компиляции программ в системах и файлы (**`ls -l`** !), которые используют эти системы. Обратите внимание, что для ошибочных программ завершающие стадии компиляции и построения выполнимой программы отсутствуют и соответствующие файлы не создаются. Файлы с ошибочными программами, также, как и тесты, должны быть заготовлены заранее. Тексты программ обязательно должны быть прокомментированы в разумном объёме (хорошим тоном считается 25% поясняющего текста) и обязательно снабжаются идентификационной информацией (ФИО студента, номер группы, № варианта, краткая формулировка задачи). Манипуляции с СП должны быть включены в протокол.

Студенты, имеющие домашний компьютер, могут выполнить следующие упражнения:

1. Проделать аналогичные действия в других доступных системах программирования на Си.
2. Установить и опробовать в домашних условиях систему программирования GNU для платформы MS Windows как наиболее простой аналог лабораторной среды. Это поможет в дальнейшем производить предварительную домашнюю отладку учебных заданий.
3. Студентам, имеющим опыт установки и администрирования различных операционных систем, целесообразно использовать СП GNU в среде UNIX (Linux, Free/Open/Net/Dragonfly BSD, MacOS X) как более полный аналог лабораторной среды. При отладке программ на различных платформах следует учесть, что аварийное завершение программ на процессорах разной архитектуры может выглядеть по-разному. Рекомендуется также ознакомиться с IDE-средой для GNU-компиляторов на платформе Intel (RHIDE).

Замечание. Использовать для домашней отладки замечательные системы программирования фирмы Borland следует с осторожностью, поскольку они работают на другой программно-аппаратной платформе и имеют некоторые отклонения от стандарта языка (не только расширения, но и сужения стандарта языка, а также нестандартизированную библиотеку).

Литература

[1] Гриффитс А. GCC. Настольная книга пользователей, программистов и системных администраторов: Пер. с англ. –К.: ООО «ТИД «ДС», 2004. –624 с. (ISBN 966-7992-34-9).

[2] Справочник MSDN. http://msdn.microsoft.com/ru-ru/library/610ecb4h.aspx

Приложение. Сценарий работы при программировании на языке Си в среде MS Visual Studio 2010 [2].

По сути, он мало отличается от GNU C.

1. **Запуск интерпретатора командной строки Microsoft Visual Studio 2010**.
Откройте окно Visual Studio Command Prompt (2010). Для этого нажмите кнопку «Пуск», а затем «Все программы» → «Microsoft Visual Studio 2010» → «Visual Studio Tools» → « Visual Studio Command Prompt (2010)».

2**. Запуск редактора текстов.** В командной строке введите команду:

```
notepad lab8.c
```

3. **Ввод текста программы.** В окне блокнота наберите текст вашей программы. В меню «Файл» выберите пункт «Сохранить» и закройте блокнот.

4. **Компиляция программы.** Теперь введите с консоли команду:

```
cl lab8.c
```

В результате будет получен выполнимый файл lab8.exe

5. **Запуск программы.** Для выполнения программы введите:

```
lab8.exe
```

Вопросы к зачёту по ЛР № 8 и 10.

Часть IV. СИСТЕМЫ ПРОГРАММИРОВАНИЯ

1. Определение, структура и состав СП.
2. Языковые процессоры: трансляторы (компиляторы) и интерпретаторы.
3. Языковая среда.
4. Краткие сведения о GNU проекте. Особенности СП GNU Compiler Collection C.
5. Компиляция программ.
6. Редактирование связей и создание выполнимых программ.
7. Компиляция с созданием выполнимой программы.
8. Задание флагов СП в командной строке и в тексте программы. Флаги «только компиляция», «компиляция-линкидж», «линкидж», «именование выполнимой программы», «подключение библиотек», «уровень оптимизации», «проверка стандарта языка», «версия СП», «трассировка работы компонент СП».
9. Флаги «генерация защитного кода», «генерация отладочного кода».
10. Файлы СП С.
11. Понятия отладки и тестирования.
12. Отладка путем ручной прокрутки.
13. Отладка средствами стандарта языка.
14. Отладчик СП GNU в среде ОС UNIX.
15. Команды отладчика.
16. Сценарий работы в СП MS Visual Studio.
17. Особенности отладчика СП MS Visual Studio.

Задание подготовили: проф. Зайцев В.Е., ст.преп. Калинин А.Л., ст. преп. Лебедев А.В., прогр. Измайлов А.А., Миронов Е.С. и Перетягин И.А.