ИНФОРМАТИКА

2005 **июль-сентябрь** **№ 3**

УДК 681.142.2

Д.И. Черемисинов

РАБОТА С КЛАСТЕРНЫМ КОМПЬЮТЕРОМ ИЗ WINDOWS

Рассматривается проблема подготовки программы к выполнению на мультипроцессорной системе кластерного типа. При разработке программ для кластерного компьютера применяется технология, основанная на использовании удаленного терминала. Рассматривается ситуация, когда таким удаленным терминалом является компьютер с операционной системой Windows. Предлагается набор инструментальных средств, позволяющий выполнять задачи редактирования текста, компиляции программы и запуска программы на кластере. Достоинством предлагаемого способа подготовки программы к выполнению является возможность максимального использования опыта работы программиста в Windows.

Введение

Под кластером понимают совокупность процессоров, объединенных компьютерной сетью и предназначенных для решения одной задачи, как правило, большой вычислительной сложности. Кластер – довольно сложный объект и для работы на нем нужно иметь знания по сетям, устройству компьютеров, операционным системам, специальному программному обеспечению. На абсолютном большинстве кластеров в качестве операционной системы используется Linux. Можно предположить, что каждый, кому приходится программировать для кластерного компьютера, должен установить Linux в качестве рабочей системы на своем компьютере, забыть все свои навыки, связанные с работой в Windows, и научиться работать в UNIX. Между тем в статье не призывается переходить на Linux, как раз наоборот: эта статья предназначена для тех, кто разрабатывал программы для Windows на C++ и намерен пользоваться привычными технологиями и инструментами при разработке программы для кластера. Естественно, что при разработке программы для кластера хотелось бы изучать как можно меньше технических деталей, связанных с управлением операционной системой при редактировании программы, компиляции, запуске на выполнение, и сосредоточиться на разработке параллельного алгоритма. Этого можно достичь, если использовать компьютер с Windows как удаленный терминал кластерного компьютера. Однако минимальные сведения об устройстве операционных систем на основе UNIX все же необходимы. Поэтому эта статья не претендует на роль учебника по UNIX, ее цель состоит в том, чтобы описать самые основные действия, которые приходится выполнять в ходе процесса разработки программы, и особенности процесса подготовки программы с точки зрения программиста, использующего в основном системы с Windows.

1. Основные сведения о UNIX

Пользователи Windows и DOS привыкли, что адрес файла выглядит как *c:\dir1\dir2\filename.ext*. В UNIX путь к файлу выглядит немножко по-другому: */dir1/dir2/filename.ext*. В UNIX понятие диска отсутствует, все адреса начинаются с корня, обозначаемого символом «/». Обратите внимание, что в качестве разделителя используется прямой слэш, а не обратный. Использование обратного слэша – это очень распространенная ошибка. Путь к файлу может быть абсолютным (от корня, как показано выше) или относительным (так же, как и в DOS). Две точки, как и в DOS, означают вышестоящий каталог, одна точка – текущий. Надо учитывать, что большие и маленькие буквы в UNIX различаются: *file.txt* и *File.txt* – это два разных файла, которые вполне могут лежать в одной директории.

В отличие от Windows файловая система UNIX представляет собой граф, а не дерево. Кроме каталогов и файлов файловая система UNIX имеет еще один объект, называемый ссылкой. Ссылки связывают пары любых объектов в файловой системе. Делать ссылки на директории удобно, чтобы давать короткие имена или «объединять» директории, находящиеся в разных местах.

Одна из проблем, с которой регулярно сталкиваются пользователи UNIX, – это права доступа к файлам. Наиболее частыми ошибками являются попытки сохранять данные в каталог, в котором у вас нет прав на запись, а также отсутствие права на выполнение программы. Есть три вида разрешений на доступ к файлам: чтение, запись и выполнение, которые присваиваются каждому файлу или каталогу. В UNIX имеется три типа разрешений: для владельца файла, для группы пользователей, к которой владелец принадлежит, и для всех остальных пользователей. Право на выполнение для каталогов дает доступ к содержимому этих каталогов. Различные комбинации прав для файлов и каталогов дают разные интересные эффекты. Например, право на чтение и запись файла не дает автоматически права на его удаление или переименование – это определяется правами каталога. Комбинация прав на выполнение и чтение для каталога позволяет получать список файлов этого каталога, а права на запись и выполнение позволяют удалять файлы.

Самая «главная» команда в UNIX – это *man* (сокращение от MANual). Она является главной потому, что позволяет получить справочную информацию о любой команде операционной системы. Набрав *man команда*, вы получите подробную справку по этой команде, часто очень большую по объему. Одно из свойств ОС UNIX состоит в том, что практически каждая команда имеет различные уточняющие параметры и параметров этих могут быть сотни. Напрасно ожидать, что по этой команде вы попадете в систему, аналогичную *Help Windows*. В программе *man* не используется гипертекст, управление просмотром архаично: нажатие любой клавиши позволяет перейти на следующий экран справки, работают стрелки для прокрутки вверх и вниз, а если где-то в середине справки вы нашли то, что искали, то до конца проматывать не обязательно – для выхода достаточно нажать «q» на клавиатуре. Большинство команд также понимают параметр *--help* (два минуса и слово help без пробелов) и выдают краткую справку о своем назначении и параметрах.

В целом работа с консолью в UNIX (и в Linux также) способна вызвать панику у пользователя Windows. Кажется, что время повернуло вспять, и вы оказались на 20 лет назад, когда командная строка была последним достижением в управлении компьютером.

2. Доступ к кластерному компьютеру

Чтобы иметь возможность работать с кластером, нужно быть зарегистрированным пользователем головной машины кластера. Это можно сделать самостоятельно, если иметь доступ к идентификационной информации суперпользователя кластера. Обычно регистрацию нового пользователя выполняет системный администратор кластера. В результате регистрации пользователю выделяется место для размещения своих данных на головной машине кластера и сообщаются логин и пароль для доступа. Логин (условное имя пользователя) – это алфавитно-цифровая строка символов, используемая для идентификации пользователя в операционной системе.

Ваш компьютер с Windows должен быть подключен к той же сети, что и головная машина кластера. Для интерактивной работы с кластером и обмена файлами требуется использовать протокол с шифрованием информации SSH (Secure SHell). Если использовать протокол telnet, в котором пароли и команды передаются в открытом виде, то ваш пароль может быть перехвачен при передаче по сети и использован для несанкционированного доступа на кластер посторонних. Естественно, что при использовании посторонними ваших пароля и логина вход на кластер будет запрещен и для вас тоже (на какое-то время). Вы даже можете потерять свои файлы или другую конфиденциальную информацию. Для подключения пользователей Windows потребуется терминальная программа доступа к кластеру на вашей машине, использующая протокол SSH. Ее необходимо установить и настроить.

Очень удобно использовать программу PuTTY, которую можно взять по следующему адресу: *ftp://linux4u.jinr.ru/pub/win9x/crypt/putty-0.53b/x86/putty.exe*

Запишите ее во вновь созданную папку на своей машине. Установите на нее ярлык на своем рабочем столе. Вызовите *putty.exe* и настройте ее на соединение на 22-м порту по умолчанию и по IP-адресу, который сообщит администратор кластера. После запуска PuTTY запросит ваши логин и пароль. В случае успешного подключения к кластеру вы увидите приглаше-

ние командной строки. Это означает, что можно работать с кластером, выполняя команды операционной системы.

Программа PuTTY представляет собой сервис терминала, т. е. дает возможность вводить информацию через клавиатуру и выводить ее на экран дисплея. Если ваши данные – в первую очередь тексты программы – уже находятся на кластере, PuTTY достаточно для дальнейшей работы. Если же ваши данные находятся на вашем компьютере, то необходима программа, обеспечивающая транспортировку данных с вашего компьютера на кластер и наоборот. В принципе, для этого предназначена имеющаяся в Windows программа *ftp*. Однако использовать для транспортировки данных незащищенный протокол FTP, в котором данные передаются в открытом виде, нельзя по соображениям безопасности.

Для организации транспортировки данных с использованием защищенного канала передачи удобно использовать программу SecureFX. SecureFX – это клиентская программа протокола SHH для зашифрованной передачи файлов с широкими возможностями настройки конфигурации и протоколов передачи. SecureFX поддерживает режим докачки и восстановления связи в случае ее обрыва. Эта программа распространяется за плату.

Программа SecureFX – это клиент сервиса транспорта данных, который может передавать данные как посредством традиционного протокола FTP, так и через шифрующий протокол SSH2. Программа имеет простой и удобный интерфейс, позволяет создавать закладки для быстрого доступа к различным ресурсам, а также обладает внушительным списком из уже установленных закладок. Для загрузки файла на кластер можно использовать как стандартные функции copy-paste, так и drag-and-drop.

3. Использование Midnight Commander

Итак, тексты вашей программы находятся в вашем каталоге на головной машине кластера. Процедура подключения к кластеру после запуска PuTTY выполнена, и вы видите приглашение командной строки (рис. 1).

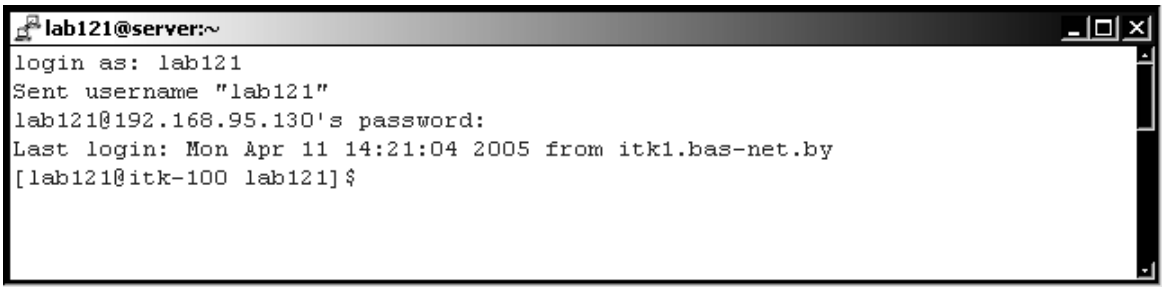

Рис.1. Образец содержимого экрана PuTTY после завершения процедуры подключения

О языке командной строки можно прочитать в любом руководстве по Linux. Обычно руководство по системам типа UNIX представляет собой толстую книгу, в которой описание команд командной строки и их параметров составляет основную часть. Эти руководства рассчитаны на фанатов UNIX, которые презирают пользователей, не умеющих работать с командной строкой. С одной стороны, командная строка – это, конечно, показатель высокого класса, но с другой – это вчерашний день в организации интерфейса пользователя.

Для пользователя Windows удобнее для управления системой через консоль использовать программу MC (Midnight Commander). В Linux последних версий она входит в комплект поставки. Для ее запуска в приглашении командной строки набираем *mc*. После этого экран PuTTY принимает вид привычного «Нортона» (рис. 2). Большинство кнопок управления работают так же, как в «Нортоне» [1] для DOS. Для тех, кто использует *Windows commander* или *FAR*, отличия в использовании *mc* связаны только с учетом особенностей файловой системы Linux. Теперь задачи поиска и просмотра файлов на кластере можно выполнять тем же способом, как и в Windows.

В тех случаях когда вам требуется немного подправить текст вашей программы, можно использовать встроенный редактор *mc* (кнопка F4). Это действительно быстрее, чем копировать файл в свой компьютер, редактировать, а потом снова выкладывать на кластер, используя SecureFX.

В случае больших изменений удобнее редактировать тексты на своей машине на привычном редакторе. Основные неудобства при редактировании на кластере связаны с невозможностью выполнения коррекций с помощью привычных операций copy-paste, трудностью переключения языка (русский/латинский) и т. п.

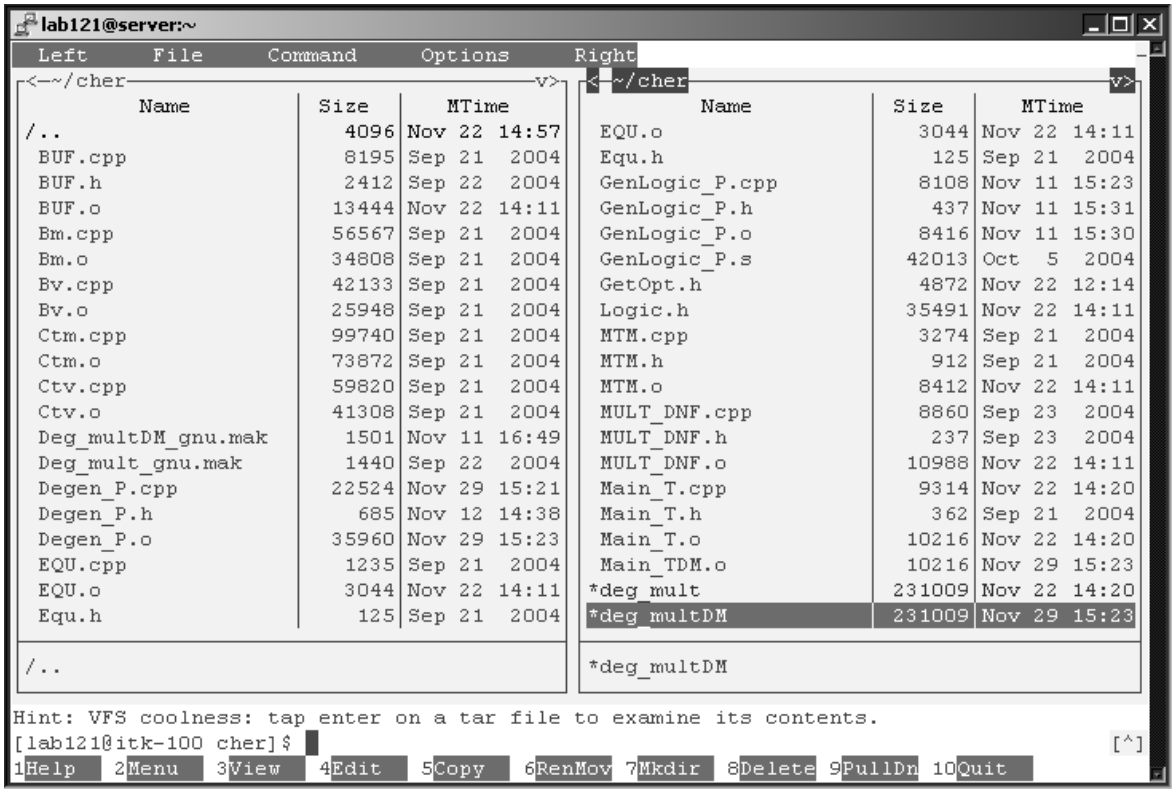

Рис. 2. Образец содержимого экрана PuTTY после запуска *mc*

Запуск программы выполняется теми же способами, что и в «Нортоне». Нужно учитывать, что исполняемые файлы программ в двоичном виде в Linux не имеют расширения. Признаком того, что файл имеет права на выполнение, является звездочка перед именем файла (рис. 2). Для установки прав доступа используется команда *chmod*, которая выбирается через меню «команды» (кнопка F9, как в «Нортоне», для доступа к главному меню).

История работы с командной строкой запоминается *mc* в ходе сеанса и сохраняется между сеансами. Для управления переходами по истории в режиме панелей служат кнопки:

Alt-p (ESC+p) – вернуть предыдущую (предыдущие) командную строку (строки);

Alt-n (ESC+n) – вернуть последующую (последующие) командную строку (строки).

Для перехода в режим экрана командной строки служит та же кнопка, что и в «Нортоне»: **Ctrl-O** – погасить панели и посмотреть, что находится под ними (работает только в Linux). Здесь же можно вводить команды командной строки непосредственно. Программы, обрабатывающие команды, носят в Linux общее название shell (командный интерпретатор). Shell тоже запоминает историю работы, однако для перехода по этой истории используются кнопки стрелок вверх и вниз. **Ctrl-O** позволяет также вернуться в режим панелей, как и в «Нортоне». История, запомненная командным интерпретатором, может не сохраняться между сеансами. Файл истории имеет фиксированный размер и запоминает историю по принципу кольца.

Если при включенных панелях не запускается через поле командной строки ни одна команда и выдается ошибка «The shell is already running a command», это значит, что нужно освободить командный интерпретатор. Погасите снова панели и нажмите ENTER не менее двух раз.

Полезные действия, которые *mc* выполняет за одно нажатие кнопок:
Shift-F3 – просмотр файла (raw – без учета расширения);
Shift-F4 – создать новый файл;
F12 – Save as.

Одна из проблем, часто вызывающих панику у пользователей, – это аварийное прекращение работы запущенной программы по инициативе пользователя. Для большинства программ поможет нажатие кнопок Ctrl-C. Перед этим нужно перейти в режим экрана командного интерпретатора, т. е. закрыть панели *mc*. Закрытие консоли на удаленной машине (выход из программы PuTTY), вообще говоря, программу не останавливает. Подробнее об этом будет написано в следующем разделе.

4. Режимы запуска программ

Для запуска программы можно просто в поле командной строки набрать ее имя, нужные параметры и нажать ввод. Сложность заключается в том, что, пока такая программа работает, пользователь ничего делать не может – надо ждать ее завершения. В случае если программа работает долго, для запуска ее в фоновом режиме надо после имени файла поставить символ «&». Посмотреть список фоновых программ, которые вы запустили, можно командой *jobs*. В полученном списке все команды будут пронумерованы, и если какую-то из них нужно перевести в обычный режим выполнения, то следует выполнить команду, например, *fg 1*. Для того чтобы перевести в фон запущенную программу, можно использовать команду *bg*. Названия команд *fg* и *bg* – это сокращения слов foreground и background. Если программа уже работает, то команду *bg* ввести не получится, но выход есть: сначала надо будет нажать кнопки Ctrl-Z (после чего программа приостановится), а потом уже воспользоваться *jobs* и *bg*. Режим запуска программ через поле командной строки *mc* зависит от настройки *mc* (обычно в фоновом режиме).

У фоновых программ есть свои сложности: если вы выйдете из системы (закрыв программу PuTTY), то программа будет «убита», т. е. ее выполнение будет прекращено. В фоновом режиме удобно запускать такие программы, которые считают довольно долго. В этом случае пригодится команда *nohup* (запуск команд в режиме игнорирования сигналов прерывания и завершения), которая предотвращает остановку выполнения программы при выходе из сеанса. Если выполнить команду *nohup myverylongprogram &*, то можно спокойно отключаться (закрыв программу PuTTY) и, заглянув на сервер через часок-другой (снова запустив PuTTY), увидеть результаты работы программы.

Для того чтобы узнать, какие программы сейчас запущены, используется команда *ps*. У нее довольно много параметров, указывающих, какими именно процессами вы интересуетесь и в каком формате надо показывать результаты. Полезными параметрами являются «a» (показывать не только ваши процессы, но и других пользователей); «l» (выводить более подробную информацию о каждом процессе) и «w» (не обрезать строку на 80-м символе).

У каждого процесса есть PID – Process ID. Если нужно прекратить выполнение какой-то задачи («убить процесс» в терминологии UNIX), то для этого служит команда *kill*. В качестве параметра к ней указывается PID того процесса, к которому применяется *kill*. На самом деле команда *kill* не останавливает процесс мгновенно, а передает ему особый сигнал, на который процесс реагируют остановкой.

Как было сказано ранее, команда *ps* выводит список процессов, запущенных в данный момент времени. Список этот длинный, и для того чтобы дать понять, насколько загружен кластер, команда *ps* мало пригодна. В отличие от *ps*, команда *top* выводит и обновляет в реальном времени список запущенных процессов, причем сортирует их по степени использования процессорных ресурсов. Посредством этой команды удобно определять, какая из программ является наиболее ресурсоемкой и как себя чувствует сервер в целом – в верхней строке команда *top* выводит справочную информацию о состоянии сервера: загрузку памяти, файла подкачки и процессора, разбивку загрузки между системными и пользовательскими программами, количество выполняющихся процессов и т. д.

5. Подготовка программы к компиляции

В операционной системе Linux имеются среды для разработки программ на C++, которые предоставляют программисту удобства, похожие на возможности среды MSVC в Windows, однако работа в этих системах значительно отличается от работы в среде MSVC. Один из крупных недостатков использования для доступа к кластеру компьютера с Windows состоит в том, что в этом случае приходится ограничиваться теми ресурсами Linux, которые доступны через консоль. Поэтому через удаленный терминал программы PuTTY в принципе нельзя использовать программы Linux с графическим интерфейсом. С другой стороны, использование среды для разработки программ из Linux означает девальвацию навыков, приобретенных в MSVC.

Возможна стратегия разработки на основе концепции переносимого кода, которая обеспечивает независимость разрабатываемой программы от среды разработки. Довольно удобен вариант этой стратегии, позволяющий в какой-то мере обеспечить применение знаний среды MSVC, когда последовательный прототип параллельной программы разрабатывается в среде MSVC с учетом требования переносимости кода. Переносимость кода означает использование архитектуры консольной программы в MSVC. Этот код преобразуется в параллельную программу, которая предназначена для выполнения на кластере. Для этого тексты программы должны быть откомпилированы на Linux.

На головной машине кластера всегда имеется компилятор C++, который управляется средствами командной строки. Без такого компилятора не может существовать ни одна система UNIX. Дело в том, что между разными машинами, даже имеющими один и тот же тип UNIX, не существует совместимости программ на уровне двоичных кодов. Для машин с Linux существует совместимость программ на уровне двоичных кодов при одинаковой версии ядра операционной системы. Таким образом, установка любой программы (за исключением программ, которые выполняются интерпретаторами) в UNIX обычно требует компиляции ее исходного кода.

Задание на компиляцию программы с использование компилятора C++ Linux представляет собой управляющий файл для программы *make*. Таким образом, для компиляции программы, кроме ее текстов на C++, нужно иметь файл задания для *make*. Среда MSVC позволяет экспортировать файл проекта в формате *make*. Однако файл, экспортированный из MSVC, мало пригоден для управления компиляцией на Linux, так как требует такой ручной переделки, что проще его написать заново с нуля. С другой стороны, сегодня из опытных программистов мало кто помнит, как написать управляющий файл для *make*, а молодые программисты просто не знают, что такое программа *make*.

Можно компилировать без задания для *make*, вызывая компилятор C++ непосредственно из командной строки, но в этом случае текст программы должен выглядеть как один файл, а все остальные части текста должны подключаться явно с помощью директив *#include*. Это очень неудобно по многим причинам. Одна из существенных причин использования *make* заключается в том, что она предоставляет возможность ускорить компиляцию программы, состоящей из нескольких файлов. Программа *make* позволяет использовать объектные файлы и выполняет перекомпиляцию только тех файлов с исходным текстом, которые были изменены. Кроме того, при использовании *make* шаги сборки программы можно и не знать.

Формат файла для *make* довольно сложен, и построение задания требует значительной работы: нужно помнить, как одни файлы зависят от других, и указать все файлы, которые потребуются при компиляции вашей программы. Поэтому часто приходится указывать в управляющем файле для *make* довольно много данных. Если при компиляции требуется использовать несколько различных инструментов (например, редактор связей кроме компилятора), то нужно предусмотреть управление каждым из них. Желающие строить файлы для *make* вручную могут прочитать об этом по адресу *http://adm.jinr.ru/doc/gnumake/*

Существуют программы, позволяющие сгенерировать управляющий файл автоматически. С этой целью удобно использовать программу *genmake*, которую можно взять по следующему адресу: *http://www.robertnz.net/genmake.htm*

Проект, для которого генерируется управляющий файл для *make*, должен быть подготовлен. Для этого в каждый *.h* файл проекта нужно включить специальные директивы в формате

комментария. В этих директивах указываются имена *.cpp* файлов, в которых содержатся определения объектов, декларированных в соответствующем *.h* файле. Формат директивы для genmake следующий:

```
// body file: liststr.cpp
```

Для *genmake* указывается название *.cpp* файла, содержащего функцию *main* программы. Программа *genmake* просматривает этот файл для нахождения директив *#include* и включения *.h* файлов. Найденные *.h* файлы просматриваются с целью поиска директив *genmake*, в которых указаны .cpp файлы, содержащие определения объектов, декларированных в соответствующем *.h* файле. Таким образом, разыскиваются все требуемые файлы и формируются зависимости между ними. Предполагается, что все файлы проекта содержатся в одном каталоге.

Например, запуск *genmake* командой

```
genmake -g tmt > tmt_gnu.mak
```

означает, что строится файл *tmt_gnu.mak* для проекта, в котором функция *main* программы содержится в файле *tmt.cpp*. Флаг *–g* означает, что для построения файла для *make* будет использован префиксный файл *pre_g.txt*. Этот префиксный файл для подключения библиотеки MPI должен иметь следующий вид:

```
# MPI settings
ifndef USE_MPICH
   USE_SCAMPI     =;
endif

ifdef USE_SCAMPI
   MPI_HOME   = /opt/scali
   MPI_LDLIBS = -lmpi -lpthread
endif

ifdef USE_MPICH
   MPI_HOME   = /opt/scali/contrib/mpich
   MPI_LDLIBS = -lmpich
endif

# Compiler settings
ifndef CXX
   CXX     = g++
   OPT     = -O3
endif
CXXFLAGS     = -D_REENTRANT -I$(MPI_HOME)/include $(OPT) $(CC_MACH_FLAGS)

%.o:                %.cpp
        $(CXX) $(CXXFLAGS) -c $*.cpp
```

К сожалению, сгенерированный *genmake* файл требует ручной доработки. Его нужно открыть в любом редакторе, найти строчки, похожие на

```
main_t:              $(main_t_obj)
        $(CXX) -o $@ $(main_t_obj) -L. -lm
```

и заменить в последней строке часть, начинающуюся с `-L.`, до конца строки на

```
-L$(MPI_HOME)/lib $(MPI_LDLIBS) $(MACH_LDFLAGS)
```

так, чтобы она выглядела следующим образом:

```
        $(CXX) -o $@ $(main_t_obj) -L$(MPI_HOME)/lib $(MPI_LDLIBS)
$(MACH_LDFLAGS)
```

Компиляция на Linux с использованием подготовленного файла *tmt_gnu.mak* выполняется командой

```
[lab121@itk-100 lab121]$ make -f tmt_gnu.mak
```

где [lab121@itk-100 lab121]$ – это приглашение командного интерпретатора для ввода командной строки. Если компиляция прошла успешно, то в директории с программой должен появиться файл программы.

Для запуска параллельной программы, например с именем *TMT* на 21 процессоре, требуется выполнить следующую команду:

```
[lab121@itk-100 lab121]$ /opt/scali/bin/mpirun -np 21 TMT
```

6. Проблема русского языка

Принципиальное отличие ситуации с русским языком в Linux от других языков связано с тем, что на сегодняшний день существует несколько кодировок для русского языка: Windows 1251, KOI-8r, ISO 8859-5 и др. По умолчанию кодовой страницей русского языка для Linux является KOI-8r, а в Windows кодовая страница русского языка – это Windows 1251. Различие в стандартах кодировки ведет к тому, что тексты программ, подготовленные для компиляции в Windows, будут неправильно выглядеть при просмотре на Linux. Но проблема не только в просмотре текста программы, через консоль очень трудно выполнить переключение раскладки клавиатуры на русский язык при редактировании. Это главная причина, по которой редактирование текстов программ удобнее выполнять на удаленном компьютере пользователя. С точки зрения удобства редактирования текста программы удобнее всего в качестве редактора использовать среду MSVC. Ее редактор особенно незаменим, когда необходимо анализировать структуру вложенности скобок. С другой стороны, для правильной работы с русским языком редактор должен обеспечивать возможность преобразования кодировки Windows 1251 в KOI-8r и наоборот. Эту возможность можно обеспечить в среде MSVC с помощью специально разработанного скрипта.

С точки зрения возможностей просмотра и редактирования текстов программ довольно удобен текстовый редактор Aditor, который можно взять по адресу *http://nrd.pnpi.spb.ru/UseSoft/tools/Aditor/aditor.htm*. Основное его достоинство (ради чего он, похоже, и создавался) – полная поддержка если не всех, то почти всех кодировок русского алфавита (KOI, Win, DOS, ISO, Mac). «Полная» поддержка означает то, что можно просматривать и редактировать файлы в любой из этих кодировок; при открытии файла его кодировка определяется автоматически и почти всегда безошибочно (а при ошибке можно и поправить); можно переводить файлы из одной кодировки в другую по желанию пользователя; автоматически перекодируется Clipboard и т. д. Кроме того, в этом редакторе подсвечивается цветом синтаксис C++.

7. Технология работы

Основой организации работы с программой для кластера является каталог на вашем компьютере с текстами программы. Эти тексты отличаются от текстов прототипа для Windows не только параллельностью, но и тем, что они перекодированы в KOI-8r (с помощью Aditor) и в *.h* файлы включены директивы *genmake*. Первым этапом разработки программы для кластерного компьютера является создание и заполнение такого каталога. Затем с помощью программы *genmake* нужно построить *make*-файл и исправить его вручную, как описано в разд. 5. Далее этот каталог с данными сопоставляется с каталогом в вашей области на кластере. Данные из каталога на вашей машине транспортируются в каталог на кластере (с помощью SecureFX). На этом подготовительная работа для компиляции и запуска программы заканчивается.

Далее каталог на кластере служит зеркалом каталога на компьютере пользователя. Файлы, с которыми идет работа, удобно держать открытыми в программе Aditor. Для компиляции и выполнения программы нужно запустить PuTTY и в нем Midnight Commander. Если при компиляции возникли ошибки, номера строк программы и диагностику ошибок можно посмотреть под панелями *mc*. Используя открытые в редакторе Aditor файлы и эти номера, можно найти соответствующие строки кода программы и устранить ошибки. После редактирования кода нужно сохранить исправления (не закрывая Aditor) и с помощью SecureFX транспортировать исправленные файлы в каталог на кластере. После обновления каталога на кластере (содержимое каталога на кластере должно быть идентично содержимому каталога на компьютере пользователя) нужно повторить компиляцию.

Если компиляция прошла успешно, нужно проследить, чтобы двоичный файл программы появился на панели *mc* и был отмечен звездочкой. Теперь можно запускать этот файл на выполнение. Результаты работы можно посмотреть под панелями *mc*.

Заключение

Статья содержит рекомендации по созданию комплекса программных средств, обеспечивающих выполнение типовых операций по подготовке к выполнению и запуску программ для

кластера, головная машина которого представляет собой UNIX-сервер, с помощью удаленного компьютера с Windows. Предлагаемая конфигурация программных средств не является единственной из возможных. Однако использование описанной технологии позволяет программисту, имеющему навыки работы с инструментами для Windows, использовать эти знания при разработке программ для кластера. Это, конечно, дается ценой отказа от применения некоторых средств, предоставляемых рабочей системой на основе Linux. Основной недостаток предлагаемого подхода состоит в невозможности использования инструментов разработки программ Linux с графическим интерфейсом. Рекомендации для пользователей кластера, работающих на рабочих компьютерах с Linux, приведены в работе [2].

Практическое применение предлагаемой конфигурации для разработки нескольких параллельных программ, последовательные прототипы которых были построены в среде MSVC, позволило добиться того, что перенос последовательного алгоритма в Linux не требовал умения работать в данной системе и основные трудозатраты состояли в разработке и отладке параллельного алгоритма, а не в овладении комплексом инструментальных средств.

Список литературы

1. Фигурнов В.Э. IBM PC для пользователя. От начинающего до опытного. – М.: ИНФРА-М, 2002. – 640 с.

2. Руководство для пользователей LINUX кластера ЛИТ ОИЯИ / В.В. Галактионов, Т.М. Голоскокова, Н.И. Громова и др. – Дубна, 2004.

Поступила 20.04.05

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: cher@newman.bas-net.by

D.I. Cheremisinov

THE OPERATIONS WITH CLUSTER COMPUTER BY REMOTE TERMINAL IN WINDOWS

A problem of program preparations for execution in cluster multiprocessor system is considered. The development of programs for cluster computer is based on remote terminal. The situation when such remote terminal is a computer with OS Windows is considered. The toolset allowing to carry out the problems of text editing, program compilation and its start on a cluster is offered. Advantage of program preparation gives the opportunity to use at most Windows programmer experience.