Выполнение программ на учебном кластере

Хотя разработка и отладка параллельных программ возможна на локальных компьютерах, для их эффективного применения необходимо соответствующее оборудование (суперкомпьютер или компьютерный кластер). При этом следует учитывать различия в программном обеспечении, используемом на персональных компьютерах и компьютерах высокой мощности; с частности.

В данном разделе описываются средства, позволяющие организовать взаимодействие клиентского компьютера с удаленным Unix-сервером и запуск на нем параллельных программ. В качестве примера рассматривается учебный кластер ЮГИНФО Южного федерального университета, однако описываемые возможности могут быть легко адаптированы и для других вариантов Unix-систем.

Удаленная работа с Unix-сервером

Вспомогательные программы для Windows-клиента

Для удаленного подключения клиентского компьютера, работающего под управлением ОС Windows, к Unix-серверу, можно использовать свободно распространяемую *терминальную программу* PuTTY (дистрибутив программы PuTTY доступен по следующий ссылке http://www.chiark.greenend.org.uk/~sgtatham/putty/). Специальной установки программа не требует; на клиентском компьютере достаточно иметь исполняемый файл putty.exe; который можно запускать даже со съемного носителя. Использование инсталляционного файла putty-0.62-installer.exe позволяет установить на компьютере дополнительные компоненты системы PuTTY, в том числе файлы справки в различных форматах, а также автоматически создать соответствующий раздел в меню «Пуск | Программы» и ярлык на рабочем столе.

Кроме терминальной программы необходимо использовать какую-либо программу, поддерживающую *транспортный протокол передачи файлов* (ftp-протокол). В качестве подобной программы целесообразно использовать широко известный свободно распространяемый файловый менеджер FAR (официальный сайт http://www.farmanager.com/). Дополнительным аргументом в пользу знакомства с этим файловым менеджером является то, что правила работы в нем аналогичны правилам работы в файловом менеджере Midnight Commander (MC), доступном в ОС Unix.

Используя транспортный протокол, можно пересылать файлы с клиентского компьютера на Unix-сервер и обратно, а используя терминальную программу — выполнять различные действия непосредственно на сервере, в частности, компилировать программы и запускать их на выполнение.

Настройка программы PuTTY

После запуска файла putty.exe на экране появится окно настройки программы (рис. 4.1).

В разделе Session требуется указать доменное имя Unix-сервера или его IP-адрес, номер порта и тип подключения. Например, при подключении к учебному кластеру ЮГИНФО (Южный федеральный университет) надо указать следующие настройки (именно они приведены на рис. 4.1):

```
Host Name: rsudl.cc.rsu.ru
Port: 22
Connection type: SSH
```

Кроме этих обязательных настроек желательно настроить параметры, связанные с визуализацией данных в окне программы (в противном случае информация, содержащая русский текст и псевдографику, будет выводиться некорректно). Эти параметры указываются в подразделах раздела Window.

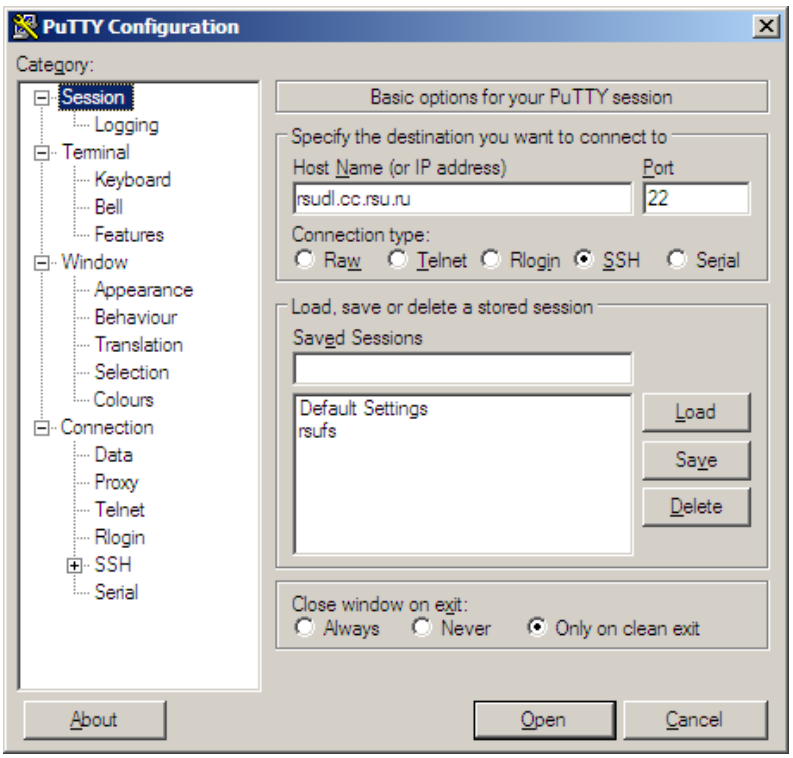

Рис. 4.1. Окно настройки программы PuTTY (раздел Session)

В подразделе «Window | Translation» следует с качестве символьного набора (Remote character set) выбрать вариант UTF-8 (рис. 4.2).

В подразделе «Window | Appearance» (рис. 4.3) следует настроить свойства шрифта, нажав кнопку [Change…] и указав в появившемся диалоговом окне в качестве набора символов вариант «Кириллический» (рис. 4.4). Можно также увеличить размер шрифта; вид шрифта (Courier New) изменять не требуется.

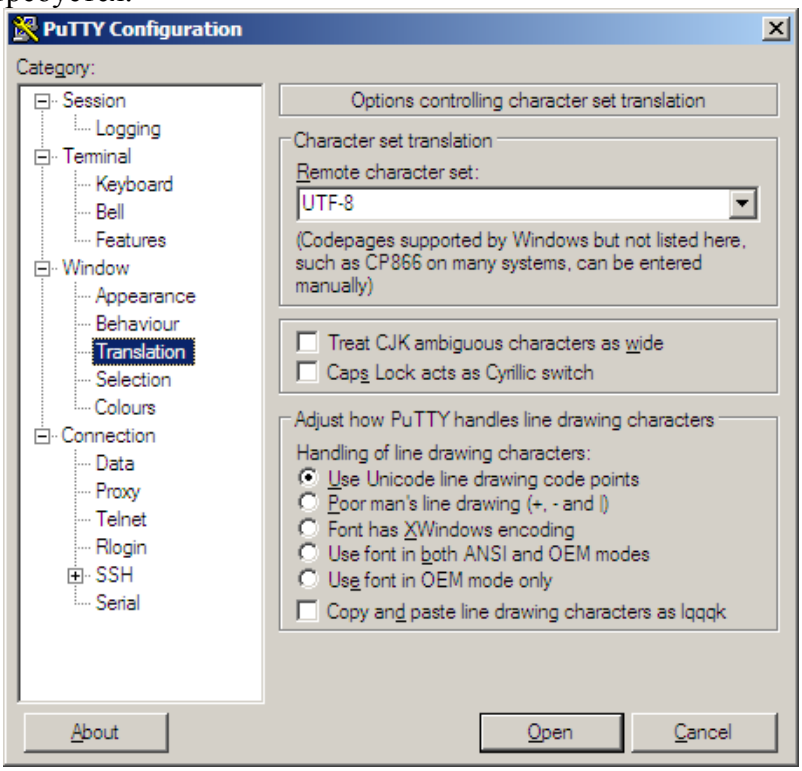

Рис. 4.2. Окно настройки программы PuTTY (раздел Translation)

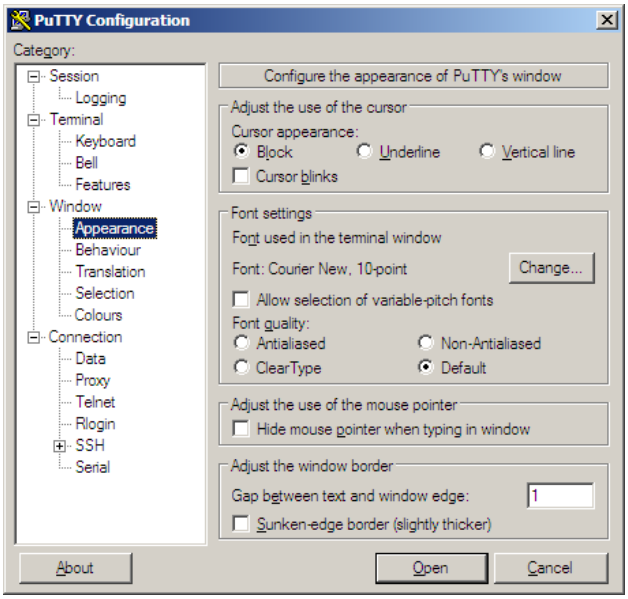

Рис. 4.3. Окно настройки программы PuTTY (раздел Appearance)

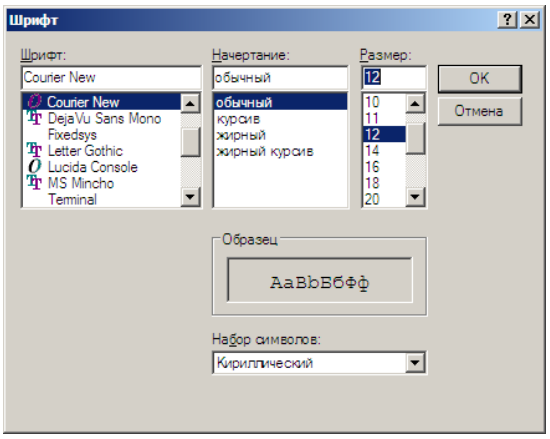

Рис. 4.4. Окно настройки шрифта

Установив указанные настройки, целесообразно вернуться в раздел Session и сохранить полученную конфигурацию под некоторым именем (например, stud). Для этого достаточно ввести имя конфигурации в поле Saved Sessions и нажать кнопку [Save]. В результате имя новой конфигурации появится в списке доступных конфигураций (см. рис. 4.5).

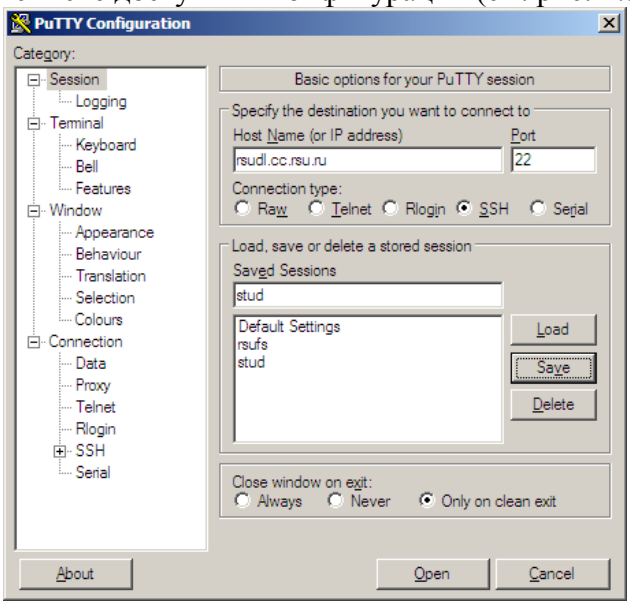

Рис. 4.5. Окно настройки программы PuTTY со списком сохраненных конфигураций

В дальнейшем для восстановления требуемой конфигурации будет достаточно выбрать ее имя из списка и нажать кнопку [Load].

Установка связи с сервером и авторизация

После настройки параметров соединения можно установить связь с Unix-сервером, нажав кнопку [Open]. Если при этом появится окно с предупреждающим сообщением, следует закрыть его, выбрав вариант «Yes».

В результате на экране появится консольное окно удаленного терминала, в котором сразу потребуется ввести логин пользователя и его пароль (введенный пароль не отображается). При правильном вводе логина и пароля система отобразит некоторые справочные сведения и выведет приглашения на ввод команд системы Unix (рис. 4.6).

При первом удаленном сеансе следует ввести стандартный пароль (например, tmppass). После первой успешной авторизации система автоматически предложит изменить стандартный пароль. Новый пароль должен содержать не менее 6 символов и включать, помимо строчных (маленьких) букв, прописные буквы, цифры или специальные символы (например, скобки). Кроме того, пароль не должен совпадать с логином или быть похожим на него.

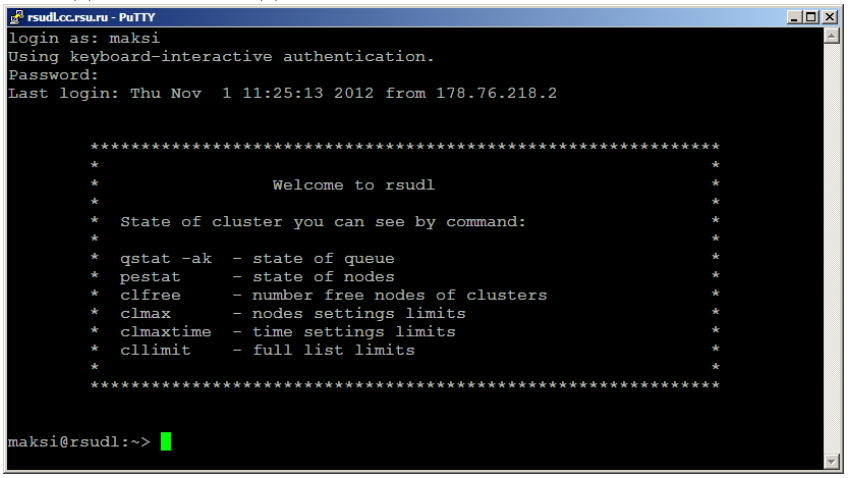

Рис. 4.6. Окно удаленного терминала

Файловый менеджер Midnight Commander

Для просмотра содержимого пользовательского каталога и стандартных действий с файлами можно сразу ввести команду mc, запускающую файловый менеджер Midnight Commander (см. рис. 4.7).

Дополнительным преимуществом данного менеджера является возможность использования мыши для выполнения команд, наличие меню и панели подсказок.

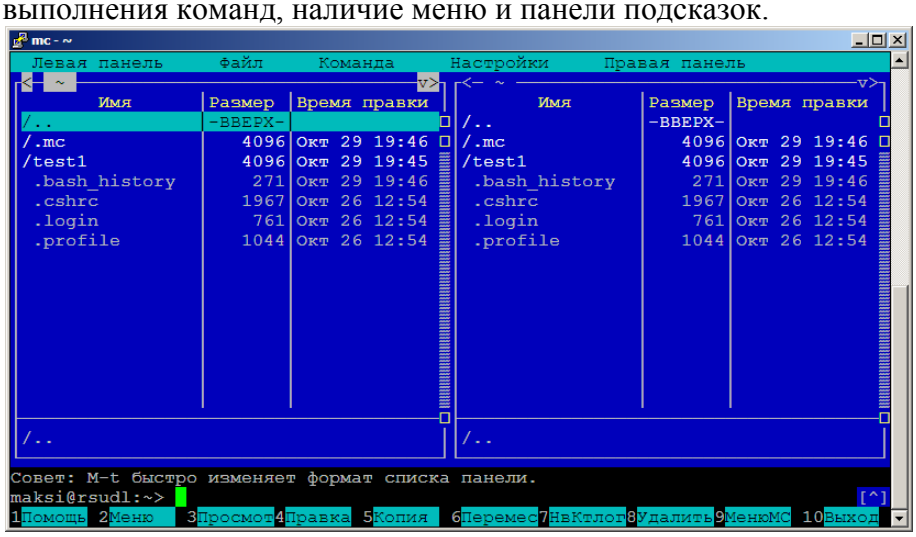

Рис. 4.7. Окно файлового менеджера Midnight Commander

С помощью функциональных клавиш F1–F10 можно выполнять наиболее распространенные команды, связанные с обработкой файлов и каталогов; подсказки к командам выводятся в нижней строке. В частности, для создания нового подкаталога предназначена клавиша F7, для удаления файла или каталога — клавиша F8 (при попытке удалить непустой каталог будет выведено предупреждение).

Для отмены какого-либо действия необходимо *дважды* нажать клавишу Esc.

Для редактирования текстовых файлов предназначена команда «Правка», связанная с клавишей F4. С помощью комбинации Shift+F4 можно создать пустой файл, набрать его текст и сохранить файл под требуемым именем (см. рис. 4.8).

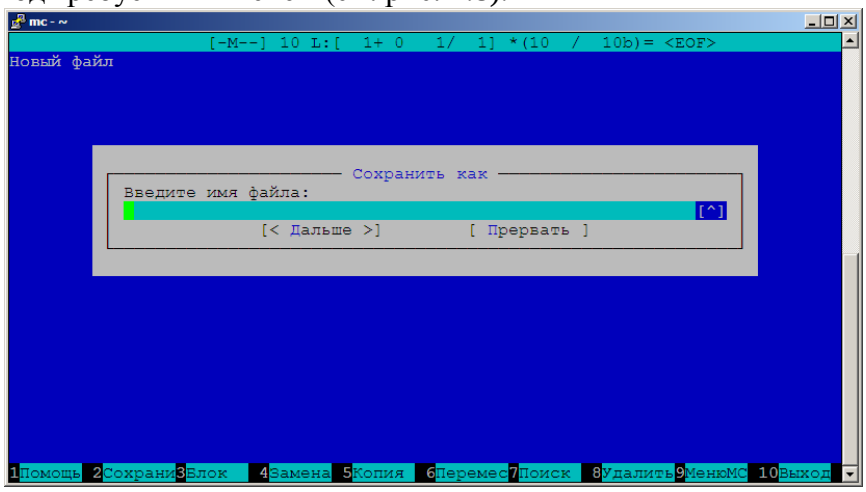

Рис. 4.8. Вид редактора программы MC с запросом на сохранение файла

Следует обратить внимание на то, что многие команды в режиме редактирования также связаны с функциональными клавишами, и подсказки к ним отображаются в нижней строке.

Для создания на сервере новых файлов имеется и другой способ, который часто оказывается более удобным: пересылка файлов по протоколу ftp (см. далее).

Выполнение команд системы Unix

Команды системы Unix можно выполнять как в менеджере Midnight Commander, так и вне его. Для временного выхода из менеджера достаточно нажать комбинацию Ctrl+O; эта же комбинация восстанавливает изображение панелей менеджера.

Выход из менеджера позволяет отобразить на экране сообщения, связанные с последними выполненными командами (в частности, сообщения об ошибках компиляции или текст, выведенный запущенной программой). Еще одно преимущество выполнения команд вне менеджера состоит в том, что все введенные команды запоминаются и могут быть последовательно выведены на экран путем нажатия клавиш со стрелками «Вверх» и «Вниз».

Использование FAR Manager для пересылки файлов

Для пересылки файлов с помощью файлового менеджера FAR следует настроить одну из его панелей на режим ftp. Для этого достаточно нажать комбинацию клавиш Alt+F1 (для настройки левой панели) или Alt+F2 (для настройки правой панели) и в появившемся списке выбрать вариант «FTP» (рис. 4.9).

На выбранной панели появится список сохраненных доменных имен и IP-адресов. Если в этом списке требуемое имя отсутствует, необходимо выполнить команду Shift+F4, позволяющую создать новое ftp-соединение. В появившемся окне (см. рис. 4.10) достаточно ввести имя сервера (в данном случае rsudl.cc.rsu.ru), а также логин (это позволит не вводить его при подключении к серверу). Пароль в окне вводить не следует (в целях безопасности); в этом случае пароль будет запрошен при попытке подключения.

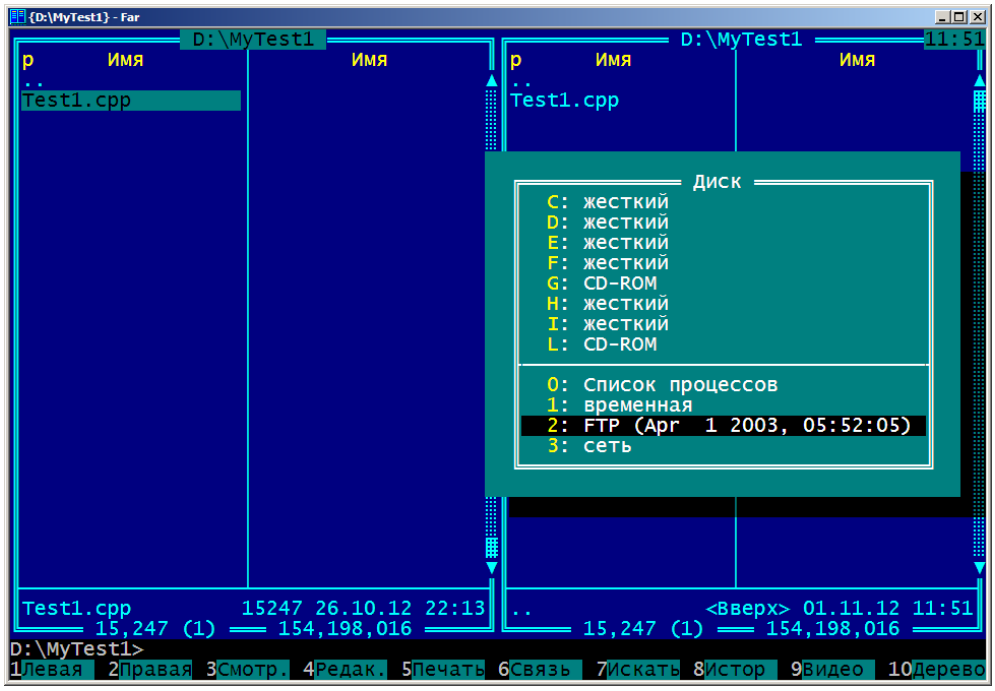

Рис. 4.9. Выбор режима ftp для панели окна программы FAR

После нажатия кнопки [Сохранить] настройки будут сохранены, и в списке FTP появится соответствующий пункт. При выборе этого пункта и нажатии клавиши [Enter] появится окно с приглашением к вводу пароля, а после правильного ввода пароля на панели будет отображено содержимое пользовательского каталога на Unix-сервере (рис. 4.11).

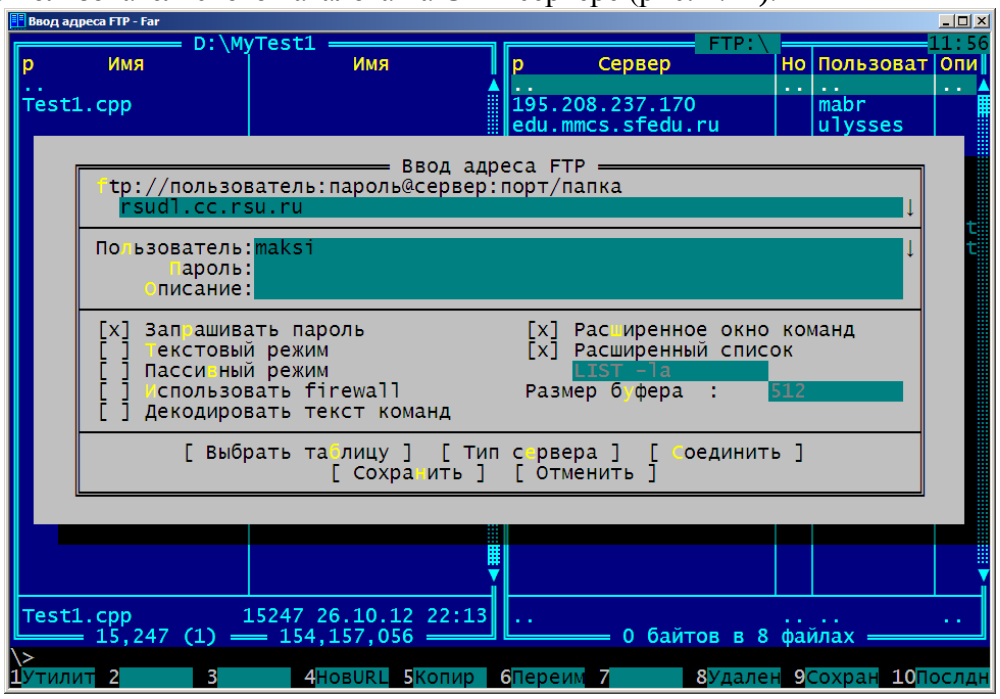

Рис. 4.10. Окно настроек ftp-соединения

Теперь можно выполнять действия по обмену файлами в любом направлении: от клиента к серверу и от сервера к клиенту. Копирование выполняется с помощью клавиши F5. Возможно копирование как отдельного файла, так и группы файлов; для выделения группы файлов предназначена клавиша Insert. Используя панель, связанную с сервером, можно также выполнять действия по созданию каталогов и файлов непосредственно на сервере, их редактированию, удалению и т. д.

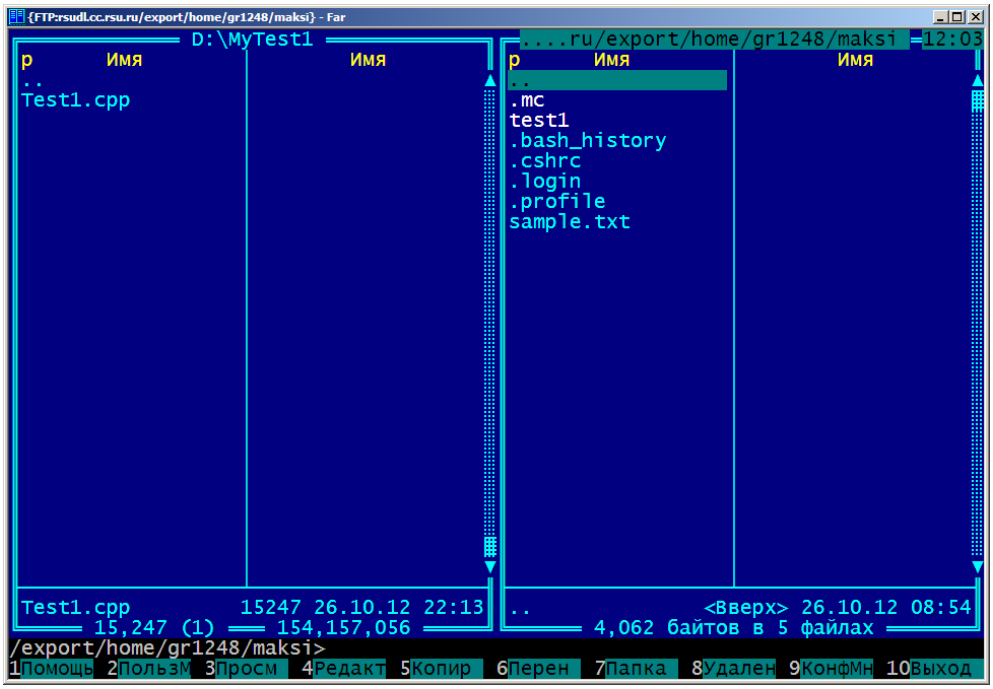

Рис. 4.11. Панель с содержимым каталога на Unix-сервере

Следует заметить, что если после подключения к серверу через протокол ftp в течение некоторого времени не выполняется никаких действий, то подключение прерывается, и для его возобновления потребуется еще раз выполнить процедуру, описанную выше.

Компиляция и запуск программ на Unix-сервере

Компиляция программ, не использующих технологию MPI

Простейшим способом компиляции программ на Unix-сервере является выполнение команды, связанной с одним из компиляторов, доступных системе. Для языков C/C++ наиболее распространенными являются компиляторы Intel, а также компиляторы пакета GCC.

В последующих примерах используются команды, связанные с C-компилятором Intel.

Для компиляции непараллельной программы или параллельной программы, использующей технологию OpenMP, применяется команда icc. В простейшем случае достаточно указать после этой команды имя компилируемого файла:

```
icc test1.cpp
```

Если в файле используются директивы OpenMP, то дополнительно следует указать опцию компиляции –openmp:

```
icc -openmp test2.cpp
```

При успешной компиляции будет создан исполняемый файл со стандартным именем a.out, для запуска которого достаточно выполнить команду

```
./a.out
```

Аналогичным образом (с указанием имени, дополненного точкой и косой чертой) должны запускаться на выполнение и другие исполняемые файлы, полученные в результате компиляции программ.

Для указания имени выходного исполняемого файла необходимо использовать опцию –o, после которой (через пробел) указывается имя исполняемого файла:

```
icc -o test1.out test1.cpp
icc -openmp -o test2.out test2.cpp
```

Если программа включает несколько исходных cpp-файлов, они указываются через пробел:

```
icc -o test1.out test1.cpp add.cpp
```

Заголовочные файлы (h-файлы) в списке файлов указывать не требуется, однако если некоторый файл подключает заголовочный файл, не являющийся стандартным, необходимо, чтобы компилятор смог найти этот файл (достаточно разместить заголовочный файл в том же каталоге, что и cpp-файлы).

Еще одной полезной опцией компиляции является опция настройки режима оптимизации –O (не следует путать эту опцию с опцией указания имени исполняемого файла, в которой используется строчная (маленькая) буква o). Имеются четыре варианта этой опции: –O1, –O2, –O3, -O. Последний вариант эквивалентен варианту –O2. Большее числовое значение соответствует более высокому уровню оптимизации, как правило, наиболее подходящим для большинства программ является уровень 2.

Для компиляторов пакета GCC предусмотрены две команды, связанные с языками C/C++:

gcc — компиляция C-программ,

g++ — компиляция программ на языке C++.

В частности, при использовании в программе потоков ввода-вывода следует использовать команду g++. Кроме того, следует учитывать, что компиляторы пакета GCC обязательно требуют, чтобы функция main имела возвращаемое значение типа int (компилятор Intel допускает, чтобы функция main имела тип void).

При компиляции OpenMP-программ с использованием компиляторов пакета GCC необходимо указывать опцию –fopenmp.

Все возможности, связанные с определением имени выходного файла, у компиляторов GCC совпадают с аналогичными возможностями компилятора Intel. Для настройки режима оптимизации у компиляторов GCC имеются три опции: –O, –O2, –O3.

Для ознакомления со всеми возможностями компилятора достаточно выполнить связанную с ним команду, указав единственную опцию --help (перед словом help указываются *два* дефиса). Для того чтобы перенаправить содержимое выводимой справочной информации в файл, следует после команды указать директиву перенаправления вывода > и имя файла, например:

```
icc --help > icchelp.txt
gcc --help > gcchelp.txt
```

Компиляция и запуск программ, использующих технологию MPI

Для компиляции программ, использующих технологию MPI, в пакете компиляторов Intel предусмотрена специальная команда mpicc. Эта команда имеет те же опции, что и команда icc (в том числе и опцию –openmp для создания так называемых *гибридных* параллельных программ), однако автоматически подключает все компоненты библиотеки MPI. Пример использования данной команды:

```
mpicc -o test3.out test3.cpp
```

Для запуска полученной программы непосредственно на сервере (без распределения процессов по различным узлам кластера) достаточно выполнить команду вида

```
mpirun -np 4 test3.out
```

После опции –np указывается количество процессов, каждый из которых запускается на сервере.

Если требуется выполнить MPI-программу на наборе узлов кластера, необходимо сформировать задание для *системы управления заданиями* PBS (Portable Batch System), установленной на кластере, и запустить это задание. Программа, запускаемая на кластере, не должна содержать команд интерактивного ввода.

Удобно занести все настройки, связанные с формируемым заданием, в специальный файл (скрипт), который затем передать в качестве параметра команде формирования задания. Приведем пример содержимого такого файла:

```
#!/bin/sh
#PBS -l walltime=00:05:00
#PBS -l nodes=5:DELLE
```

```
#PBS -q DELLE
cd $PBS_O_WORKDIR
mpirun -np 5 test3.out
```

В этом примере устанавливается максимальное время счета, равное 5 минутам, указывается число узлов кластера (5) и задается имя учебного кластера (DELLE). Команда cd с параметром $PBS_O_WORKDIR устанавливает в качестве рабочего каталога тот каталог, из которого было запущено задание.

Для гибридных программ, совместно использующих технологии OpenMP и MPI, в файле с настройками задания следует дополнительно указать количество потоков, используемых по умолчанию для каждого процесса:

```
#!/bin/sh
#PBS -l walltime=00:05:00
#PBS -l nodes=5:DELLE
#PBS -q DELLE
#PBS -v OMP_NUM_THREADS=2
cd $PBS_O_WORKDIR
mpirun -np 5 test4.out
```

При настройке конфигурации заданий следует учитывать количество узлов (рабочих станций) на кластере. Например, учебный кластер DELLE (ЮГИНФО, Южный федеральный университет) включает 6 узлов с двухъядерными процессорами. Количество процессов, указываемое с помощью опции –np, может превосходить количество заказанных узлов кластера; в этом случае на некоторых узлах будет запущено несколько процессов.

Если предположить, что скрипт с настройками задания сохранен в файле qtask3, то для запуска соответствующего задания достаточно выполнить в рабочем каталоге команду

```
qsub qtask3
```

Каждому заданию присваивается уникальный числовой идентификатор (например, 52064); значение идентификатора отображается на экране при успешном выполнении команды qsub.

В результате выполнения задания в рабочем каталоге создаются два файла, имя которых совпадает с именем скрипта, а расширения имеют вид o<идентификатор задания> и e<идентификатор задания> (в нашем примере файлы будут иметь имена qtask3.o52064 и qtask3.e52064). В файле, расширение которых начинается с буквы «o», содержится текст, выведенный всеми процессами параллельной программы; в файле с расширением «e» выводится сообщения об ошибках (т. е. текст, выведенный в поток cerr). При успешном выполнении программы файлы с расширением «e» являются пустыми.

Адаптация MPI-программ, разработанных с использованием электронного задачника PT for MPI, к выполнению на кластере

При отладке параллельных MPI-программ в среде Microsoft Visual Studio задачник Programming Taskbook for MPI позволяет запускать требуемое число процессов непосредственно из интегрированной среды, а также предоставляет удобные средства вывода данных (функции Show, ShowLine и SetPrecision), позволяющие избежать конкуренции процессов за использование устройства вывода. Заготовки для подобных программ должны создаваться в виде заготовок для решения задач из группы MPIDebug; при этом номер задачи будет определять количество процессов, используемых при выполнении параллельной программы.

Примеры использования задач группы MPIDebug для реализации параллельных MPI-программ приводятся в главах 1–3, посвященных распределенным матричным и сеточным алгоритмам, а также алгоритмам решения задачи о взаимодействии *n* тел.

Для адаптации параллельной программы, разработанной с использованием задачника, к выполнению на кластере, достаточно преобразовать ее следующим образом:

1) удалить строку

```
#include <windows.h>
```

2) заменить строку

```
#include "pt4.h"
```

на

```
#include "pt4stub.h"
```

3) заменить заголовок функции

```
void Solve()
```

на

```
int main(int argv, char** argc)
```

4) в начале функции main (прежней функции Solve) заменить фрагмент

```
    Task("MPIDebug<номер задания>");
    int flag;
    MPI_Initialized(&flag);
    if (flag == 0)
      return;
    HideTask();
```

на

```
    MPI_Init(&argv, &argc);
```

5) в конце функции main добавить фрагмент

```
    MPI_Output();
    MPI_Finalize();
    return 0;
```

6) удалить функцию WinMain в конце программы.

К преобразованной программе подключается заголовочный файл pt4stub.h — вариант файла pt4.h, содержащий объявления функций Show, ShowLine, SetPrecision, MPI_Output и обеспечивающий вывод данных в консольное окно (вместо окна задачника). При выводе выполняется группировка данных по процессам, в которых эти данные были выведены. Ниже приводится образец подобного вывода для параллельной MPI-программы, решающей задачу вычисления гравитационного взаимодействия *n* тел с помощью модели «управляющий–рабочие».

```
Process 0:
Time = 397.14 PairsCount = 4049900
-285.496803732846 7.014089107234
Process 1:
Time = 397.06 PairsCount = 4856250
Process 2:
Time = 397.02 PairsCount = 4099000
Process 3:
Time = 396.76 PairsCount = 4995250
368.910141051039 41.575105017689
Process 4:
Manager Time = 395.60
```

Полученный файл можно компилировать (совместно с файлами pt4stub.cpp и pt4stub.h) как в среде Visual Studio, так и на Unix-сервере учебного кластера. При использовании среды Visual Studio следует предварительно создать пустой проект, в который добавить указанные файлы.

Ссылки для скачивания файлов pt4stub.cpp и pt4stub.h:

http://ptaskbook.com/download/pt4stub.cpp

http://ptaskbook.com/download/pt4stub.h

Задания для самостоятельного выполнения

MPIMatrix1a. Адаптировать программу, реализованную в задании MPIMatrix1, к выполнению на кластере и протестировать ее для различного числа процессов (2, 4, 8) и различных порядков перемножаемых матриц (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.

MPIMatrix2a. Адаптировать программу, реализованную в задании MPIMatrix2, к выполнению на кластере и протестировать ее для различного числа процессов (2, 4, 8) и различных порядков перемножаемых матриц (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.

MPIMatrix3a. Адаптировать программу, реализованную в задании MPIMatrix3, к выполнению на кластере и протестировать ее для различного числа процессов (4 и 16) и различных порядков перемножаемых матриц (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.

MPIGravit1a. Адаптировать программу, реализованную в задании MPIGravit1, к выполнению на кластере и протестировать ее для различного числа рабочих процессов (2, 4, 8) и различного количества обрабатываемых тел (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.

MPIGravit2a. Адаптировать программу, реализованную в задании MPIGravit2, к выполнению на кластере и протестировать ее для различного числа процессов (2, 4, 8) и различного количества обрабатываемых тел (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.

MPIGravit3a. Адаптировать программу, реализованную в задании MPIGravit3, к выполнению на кластере и протестировать ее для различного числа процессов (2, 4, 8) и различного количества обрабатываемых тел (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.

MPIGravit4a. Адаптировать программу, реализованную в задании MPIGravit3, к выполнению на кластере и протестировать ее для различного числа процессов (2, 4, 8) и различного количества обрабатываемых тел (800 и 1600), определив в каждом случае ускорение параллельного алгоритма по сравнению с его непараллельным вариантом.