Национальная академия наук Беларуси

Государственное научное учреждение
«Объединенный институт проблем информатики
Национальной академии наук Беларуси»
(ОИПИ НАН Беларуси)

ДИСТРИБУТИВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРОМЕЖУТОЧНОГО УРОВНЯ UNICORE

Руководство оператора

2024

АННОТАЦИЯ

Настоящее руководство оператора предназначено для пользователей национальной грид-сети, содержит сведения о порядке эксплуатации клиентских приложений клиента командной строки UNICORE Commandline Client (далее UCC), графического клиента UNICORE Rich Client (далее Rich Client), реализующих набор средств для подготовки, отправки и получения результатов заданий, предназначенных для выполнения на UNICORE-сайтах национальной грид-сети, а также файлового менеджера UFM, предназначенного для передачи данных с локального компьютера клиента на хранилища UNICORE-сайтов и обратно и передачи данных между хранилищами UNICORE-сайтов.

Документ содержит сведения о назначении клиентских приложений UNICORE, о действиях пользователя при работе с заданиями, о настройках параметров данных приложений, а также о создании запросов на получение сертификата пользователя национальной грид-сети и настройке клиентских приложений UNICORE для работы в национальной грид-сети.

СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ ПРОГРАММ 4
2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММ 5
2.1 Создание запроса на получение пользовательского сертификата Удостоверяющего Центра Операционного центра национальной грид-сети 5
2.2 Создание хранилища ключей/сертификатов для работы с клиентским ПО UNICORE 6
2.3 Настройка клиента командной строки UCC 7
2.4 Настройка графического клиента UNICORE Rich Client 9
2.5 Настройка файлового менеджера UFM 13
3 ВЫПОЛНЕНИЕ ПРОГРАММЫ 14
3.1 Выполнение программы UCC 14
3.2 Выполнение программы UNICORE Rich Client 21
3.3 Выполнение программы UFM 28
3.4 Установка прикладного программного обеспечения на UNICORE-сайт 28
4 СООБЩЕНИЯ ОПЕРАТОРУ 30
4.1 Получение справки об использовании UCC 30
4.2 Получение информации о ресурсах UNICORE-сайтов 31
4.3 Режим отладки 31

1 НАЗНАЧЕНИЕ ПРОГРАММ

Клиентское ПО UNICORE включает следующие приложения:

– клиент командной строки UCC;

– графический клиент Rich Client;

– файловый менеджер UFM.

Клиентское ПО UNICORE обеспечивавает возможность подготовки, запуска и получения результатов заданий, предназначенных для выполнения на UNICORE-сайтах.

Приложения UCC и Rich Client предоставляют следующие возможности:

– отправки пользовательских заданий на выполнение на UNICORE-сайтах;

– получения текущего статуса выполнения задания;

– получения на локальный компьютер клиента результатов заданий (выходных файлов и файлов вывода stderr и stdout);

– передачи файлов с локального компьютера клиента на хранилище UNICORE-сайта и обратно.

Файловый менеджер UFM предоставляет следующие основные возможности:

– передачи файлов с локального компьютера клиента на хранилище UNICORE-сайта и обратно;

– передачи файлов между хранилищами UNICORE-сайтов с возможностью выбора протокола передачи данных;

– просмотра, редактирования, переименования и удаления файлов на UNICORE-сайте.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММ

Клиентское ПО UNICORE реализовано как Java приложения, которые поставляются как набор jar-файлов (библиотеки классов Java), а также скриптов для запуска программ.

Для выполнения клиентских приложений UNICORE рекомендуется среда выполнения OpenJDK или Oracle Java (JRE или SDK).

Для работы клиентских программ UCC, Rich Client и файлового менеджера UFM необходимо выполнить предварительные настройки.

2.1 Создание запроса на получение пользовательского сертификата Удостоверяющего Центра Операционного центра национальной грид-сети

2.1.1 Предварительные настройки

Для создания запроса на получение пользовательского сертификата Удостоверяющего Центра Операционного центра национальной грид-сети необходимо предварительно установить утилиту openssl:

– для Win32 – пакет openssl-0.9.8h-1-setup.exe:
http://gnuwin32.sourceforge.net/packages/openssl.htm

– для ОС Linux – пакет openssl, который входит в дистрибутив ОС.

2.1.2 Создание запроса на получение пользовательского сертификата

Сгенерировать запрос на получение пользовательского сертификата согласно инструкции на сайте Удостоверяющего Центра Операционного центра национальной грид-сети:
http://noc.grid.basnet.by/index.php?n=Main.Ca
и послать сгенерированный запрос и необходимую для регистрации информацию, указанную в пункте 5 инструкции на сайте администратору Удостоверяющего Центра Операционного центра национальной грид-сети:
E-mail: marina@newman.bas-net.by
Tel: +375 17 379 23 60

2.2 Создание хранилища ключей/сертификатов для работы с клиентским ПО UNICORE

После получения пользовательского сертификата из Удостоверяющего Центра Операционного центра национальной грид-сети необходимо создать хранилище ключей/сертификатов для работы с клиентским ПО UNICORE.

2.2.1 Предварительные настройки

Для создания хранилища ключей/сертификатов в формате JKS необходимо предварительно установить утилиту keytool, которая входит в пакеты OpenJDK или Oracle Java (JRE или SDK).

2.2.2 Создание хранилища ключей/сертификатов в формате JKS

а) Создать хранилище ключей/сертификатов в формате PKCS12 (в приведенном примере – `user.p12`):

```
openssl pkcs12 -export -in user.pem  -inkey userkey.pem  -out user.p12 -name
"user"
```

где

- `user.pem` – сертификат пользователя,
- `userkey.pem` – закрытый ключ, сгенерированный пользователем при создании запроса на получение сертификата,
- `user.p12` – выходной файл хранилища ключей/сертификатов в формате PKCS12.

б) Создать хранилище ключей/сертификатов в формате JKS (в приведенном примере – `user.jks`) и импортировать в него созданное в а) хранилище ключей/сертификатов в формате PKCS12 (в приведенном примере – `user.p12`):

```
keytool -importkeystore -srckeystore ./user.p12 -srcstoretype PKCS12 -
srcstorepass export_password \
-destkeystore ./user.jks -deststoretype JKS -deststorepass export_password -
srcalias "user"
```

где в параметре:

`-srcstorepass export_password`

указывается пароль для хранилища `user.p12` , который указал пользователь при его создании в качестве Export Password в а)

и в параметре:

`-deststorepass export_password`

указывается пароль для создаваемого хранилища `user.jks` (пароль для создаваемого хранилища в формате JKS должен совпадать с паролем для исходного хранилища в формате PKCS12).

Замечание:

в приведенном выше примере пароли для хранилищ указаны явно.

в) Загрузить на локальную машину пользователя корневой сертификат Удостоверяющего Центра Операционного центра национальной грид-сети `bynocgca-cacert.pem` с сайта:

http://noc.grid.basnet.by/bynocgca-cacert.pem

Например, с помощью команды:

```
wget http://noc.grid.basnet.by/bynocgca-cacert.pem
```

г) Импортировать корневой сертификат `bynocgca-cacert.pem` в хранилище ключей/сертификатов в формате JKS (в приведенном примере – `user.jks`):

```
keytool -import -trustcacerts -file ./bynocgca-cacert.pem -keystore
./user.jks -alias "NOC_CA" \
-storepass export_password -noprompt
```

где `export_password` – пароль на хранилище ключей/сертификатов `user.jks`.

2.3 Настройка клиента командной строки UCC

а) По умолчанию клиент командной строки UCC считывает настройки из файла `~/.ucc/preferences`.

Например:

`/home/user/.ucc/preferences` – для ОС Linux

`C:\Users\user\.ucc\preferences` – для ОС Windows

Для настройки клиента командной строки необходимо создать в домашнем каталоге пользователя каталог `.ucc`.

Например,

~/.ucc – для ОС Linux,

и

C:\Users\user\.ucc – для ОС Windows

в который скопировать файл настроек preferences из каталога /opt/ucc/conf.

б) Отредактировать файл настроек preferences:

1) указать полный путь к хранилищу ключей/сертификатов пользователя (в приведенном ниже примере – путь к хранилищу user.jks) и указать пароль для него,

2) указать URL реестра.

Хранилище демонстрационных ключей/сертификатов user-keystore.jks содержится в каталоге:

/opt/ucc/certs/ – для ОС Linux,

ucc\certs относительно установочного каталога UNICORE – для ОС Windows.

Пароль для хранилища демонстрационных ключей/сертификатов:

```
password=the!user
```

URL реестра демонстрационного сайта DEMO-SITE:

```
registry=https://server:8080/DEMO-SITE/services/Registry?res=default_registry
```

Для работы в национальной грид-сети необходимо указать в файле preference URL реестра национальной грид-сети:

```
registry=https://unicore.basnet.by:8080/REGISTRY/services/Registry?res=default_registry
```

Ниже приведен пример файла preference для работы грид-пользователя в национальной грид-сети.

Пример файла preferences:

```
#
# sample user preferences for UCC
#
# if not specified using the -c option, UCC checks whether a
# file '$HOME/.ucc/preferences' exists and reads preferences from it
#
keystore=C:\\Users\\user\\certs\\user.jks
#password=
registry=https://unicore.basnet.by:8080/REGISTRY/services/Registry?res=default_registry
output=.
```

Замечание: если в целях безопасности пользователь не указал в файле настроек preferences в строке:

```
#password=
```

пароль, то он будет запрошен во время соединения с реестром.

2.4 Настройка графического клиента UNICORE Rich Client

Для запуска графического клиента UNICORE Rich Client необходимо в терминале выполнить команду `richclient`. При первом запуске появится окно с требованием указать файл хранилища ключей/сертификатов (рисунок 1):

Рисунок 1 – Выбор файла хранилища ключей/сертификатов.

Демонстрационный файл хранилища ключей/сертификатов `demouser.jks` содержится в каталоге `/opt/richclient/certs/` (на Windows-системах в папке `richclient\certs` относительно установочного каталога UNICORE).

По умолчанию пароль для файла `demouser.jks` уже введен, это «321».

Выбрать файл хранилища ключей/сертификатов пользователя и ввести пароль выбранного хранилища (рисунок 2).

Рисунок 2 – Выбор файла хранилища ключей/сертификатов пользователя и ввод пароля.

В открывшемся окне программы выбрать Workbench (рисунок 3), после чего откроется главное окно программы (рисунок 4).

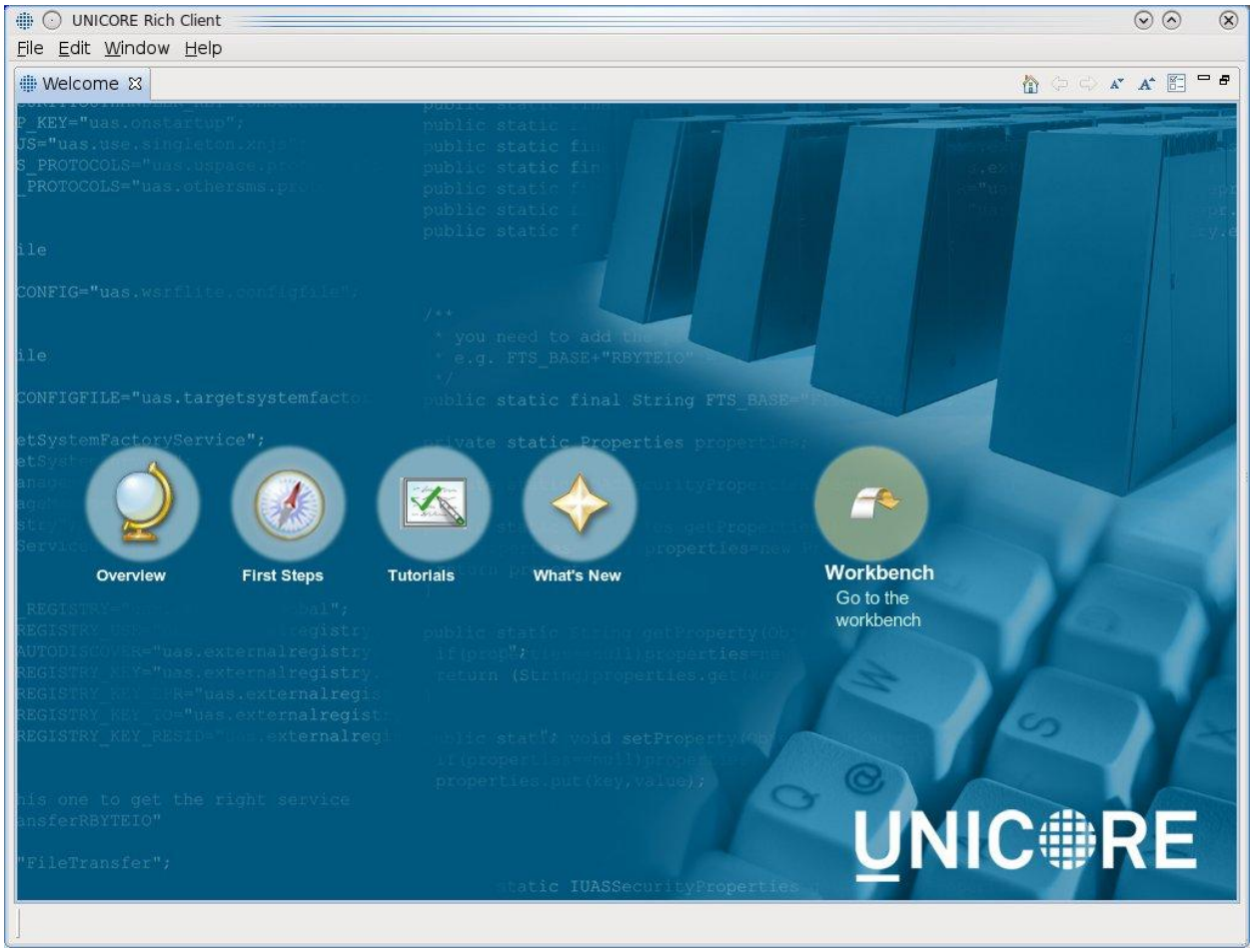

Рисунок 3 – Окно программы UNICORE Rich Client.

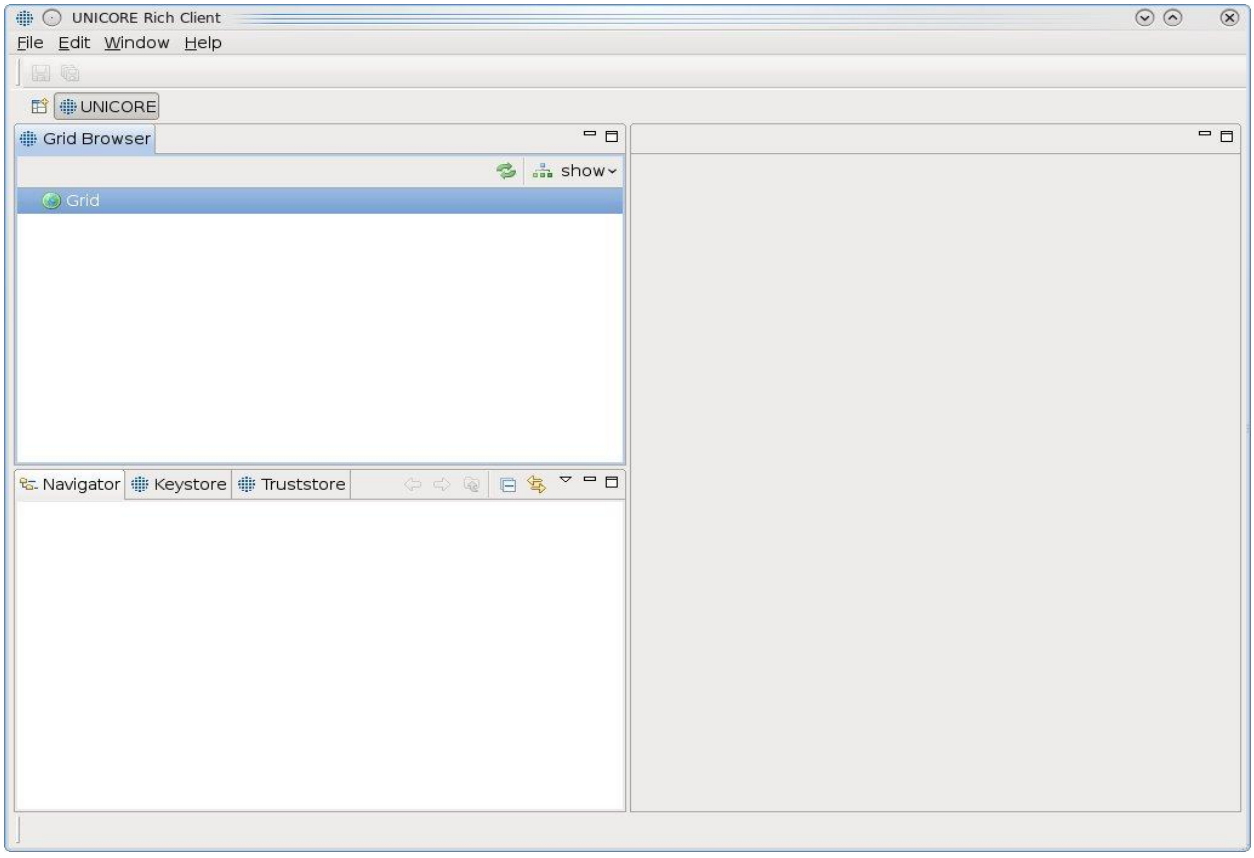

Рисунок 4 – Главное окно программы UNICORE Rich Client.

В главном окне приложения на вкладке Grid Browser выбрать в списке «Grid» и в контекстном меню выбрать «add Registry». Появится окно добавления реестра (рисунок 5), указать в строке «URL» URL национальной грид-сети:

`https://unicore.basnet.by:8080/REGISTRY/services/Registry?res=default_registry`

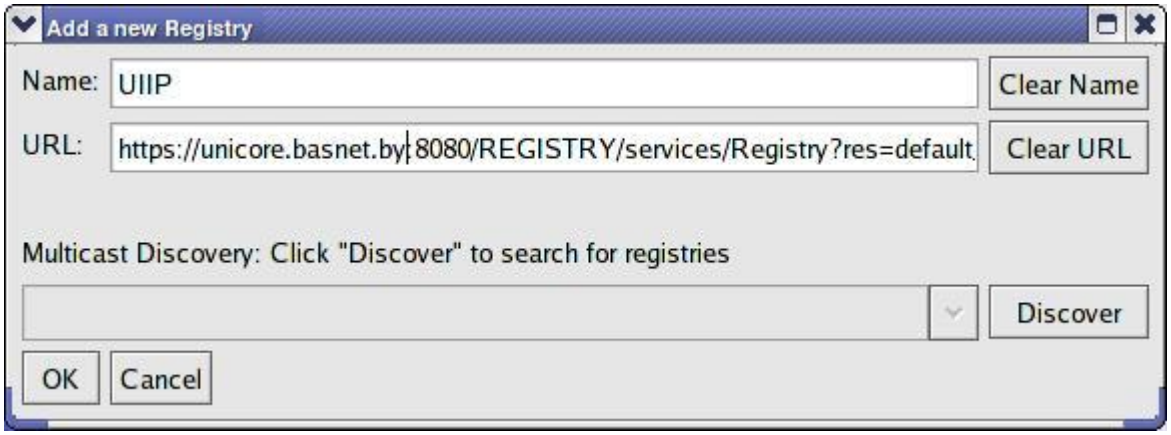

Рисунок 5 – Добавление URL реестра национальной грид-сети.

Затем в главном окне приложения на вкладке Grid Browser выбрать в списке «Grid» и в контекстном меню выбрать «refresh», после этого в окне приложения появится список UNICORE-сайтов национальной грид-сети (рисунок 6).

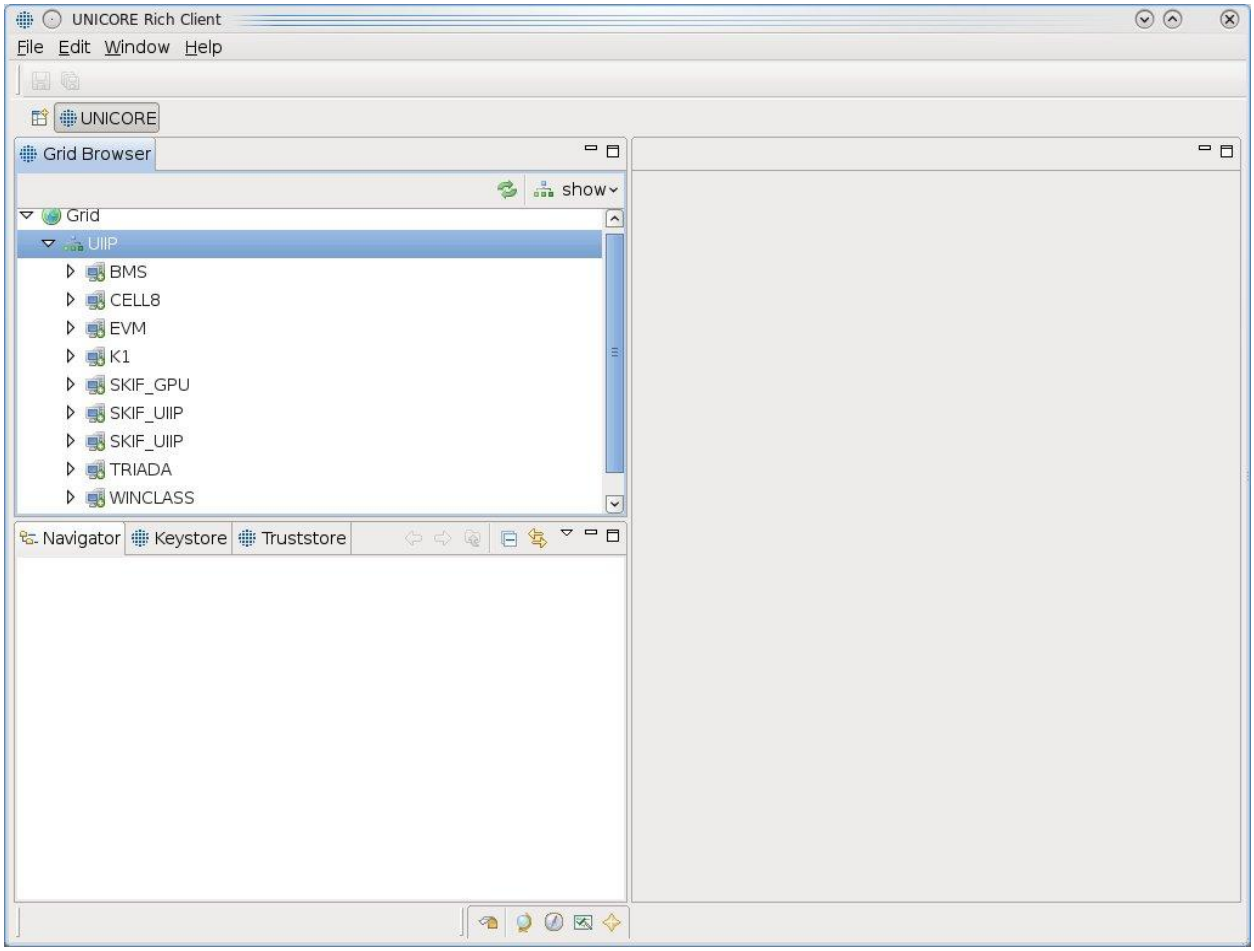

Рисунок 6 – Список UNICORE-сайтов.

2.5 Настройка файлового менеджера UFM

С помощью пункта меню Конфигурация - > Настройка вызвать окно Настройка, в котором:

а) на вкладке Сертификаты:

1) в поле Файл ключей указать хранилище ключей/сертификатов в формате JKS (в приведенном примере - user.jks)

2) В списке:

Тип файла ключей

выбрать тип jks

3) С помощью кнопки:

Ввести пароль

вызвать окно:

Ввод пароля

в котором ввести пароль для хранилища ключей/сертификатов (в приведенном примере для хранилища `user.jks`).

б) На вкладке Реестры:

1) с помощью кнопки Добавить вызвать окно:

Добавление адреса реестра

в котором ввести URL реестра.

Например,

URL реестра национальной грид-сети:

`https://unicore.basnet.by:8080/REGISTRY/services/Registry?res=default_registry`

в) Сохранить выполненные настройки с помощью кнопки:

Сохранить

Более подробную информацию по настройке файлового менеджера можно получить в руководстве оператора файлового менеджера.

3 ВЫПОЛНЕНИЕ ПРОГРАММЫ

В данном разделе подробно рассмотрена работа клиентских приложений из состава дистрибутива UNICORE, а также подготовка и запуск заданий на примере тестового задания MPI.

3.1 Выполнение программы UCC

Приложение UCC – клиент командной строки, все операции с ним выполняются в командной строке.

3.1.1 Пример тестового задания

Для тестирования выполнения грид-задания на UNICORE-сайтах используется тестовое задание для выполнения параллельной MPI-программы, вычисляющей число Пи.

Тестовое задание лежит в каталоге `/opt/ucc/samples/cpi` (в каталоге ucc\samples\cpi относительно установочного каталога для Windows-систем) Перед запуском данный каталог необходимо скопировать в свою домашнюю директорию. Задание представляет собой три файла:

- `test.u` —файл задания для клиента UCC;
- `cpi.c` — исходный текст MPI-программы;
- `script.sh` — скрипт, который выполняет компиляцию, сборку и запуск на выполнение программы на UNICORE-сайте. В случае подготовки задания на клиентском рабочем месте под ОС Windows для выполнения на UNICORE-сайте, работающего под управлением ОС Linux, необходимо учитывать разный символ конца строки в этих системах. Для преобразования текстового файла из Windows формата в Linux формат можно использовать расширенные текстовые редакторы, например, UltraEdit, или утилиту *dos2unix.exe*.

Файл `test.u` имеет следующее содержание:

```
{

Name: Test_cpi,
#
# ... or alternatively specify the executable directly
#
  Executable: "/bin/sh",
```

```
  Arguments: ["input.sh"],

#
# Import files from the local machine to the job's working directory
#

  Imports: [
    { File: "script.sh", To: "input.sh" },
    { File: "cpi.c", To: "cpi.c" },
  ],

#
# Exports files from the job's working directory to local machine
#

  Exports: [
    { File: "pi.txt", To: "pi.txt" },
  ],

#
# Resources
#
  Resources: {
    Memory: 2000000,
    Nodes: 1,
    CPUsPerNode: 1,
    Runtime: 1200,
  }
}
```

Файл `script.sh` для Linux имеет следующее содержание*:*

```
PWD=`pwd`
### List of nodes assigned  to  the job ###
cat $PBS_NODEFILE |tee pbs-nodes
NP=`wc -l pbs-nodes | awk '{print $1}'`
CC=mpicc
MPIRUN=mpirun
MPIBIN="cpi"
MPIPROG="cpi.c"
$CC -o $MPIBIN ./$MPIPROG
date
echo "Job start on $NP nodes"
$MPIRUN -v -np $NP -hostfile pbs-nodes $MPIBIN
date
```

Файл script.sh для Windows 7 x86_64 (файл `script.sh` для ОС Windows 7 расположен в каталоге `ucc\samples\cpi`).

имеет следующее содержание:

```
PWD=$PBS_O_WORKDIR
cd $PWD
### List of nodes assigned  to  the job ###
cat $PBS_NODEFILE |tee pbs-nodes
NP=`wc -l pbs-nodes | awk '{print $1}'`
MPIPROG="cpi.c"
OBJ="cpi.obj"
MPIBIN="cpi.exe"
date
echo "Job start on $NP proc"
```

```
export PATH="/cygdrive/C/Program Files/Microsoft
SDKs/Windows/v7.0/bin":"/cygdrive/C/Program Files (x86)/Microsoft Visual
Studio 9.0/VC/bin/amd64":"/cygdrive/C/Program Files/MPICH2/bin":"$PATH"


export WindowsSdk_INCLUDE="C:/Program Files/Microsoft
SDKs/Windows/v7.0/include"
export VC_INCLUDE="C:/Program Files (x86)/Microsoft Visual Studio
9.0/VC/include"
export MPICH2_INCLUDE="C:/Program Files/MPICH2/include"


export WindowsSdk_LIBPATH="C:/Program Files/Microsoft
SDKs/Windows/v7.0/lib/x64"
export VC_LIBPATH="C:/Program Files (x86)/Microsoft Visual Studio
9.0/VC/lib/amd64"
export MPICH2_LIBPATH="C:/Program Files/MPICH2/lib"


cl.exe /nologo /MT /W3 /EHsc /Ox /I "$WindowsSdk_INCLUDE" /I "$VC_INCLUDE" /I
"$MPICH2_INCLUDE" /D WIN64 /D NDEBUG /D _CONSOLE /D _MBCS /c $MPIPROG

Link.exe mpi.lib /nologo /subsystem:console /machine:x64 /out:$MPIBIN
/libpath:"$WindowsSdk_LIBPATH" /libpath:"$VC_LIBPATH"
/libpath:"$MPICH2_LIBPATH" $OBJ

date
mpiexec -n $NP -delegate –machinefile pbs-nodes "$PWD/$MPIBIN"
date
```

При запуске задания происходит загрузка на UNICORE-сайт файлов `cpi.c` и `script.sh`, затем компиляция и сборка приложения `cpi` и запуск его на выполнение на UNICORE-сайте.

3.1.2 Файл задания

3.1.2.1 Формат файла задания

Клиент командной строки UCC использует простой формат описания заданий в файле задания с расширением *.u, который позволяет указывать приложение или исполняемый файл для запуска, аргументы и переменные окружения, передачу файлов до и после выполнения задания. Формат называется JSON, и содержит разделенные запятой значения, которые могут быть простой строкой или списком значений. Строки должны быть помещены в кавычки. Для комментирования строк необходимо использовать символ «#», как в скриптах. Несколько примеров файлов заданий находятся в каталоге /opt/ucc/samples/.

В файле описания заданий можно указывать исполняемые файлы, аргументы и переменные окружения. Например:

```
    Executable: "sh",
```

```
        Arguments: ["script.sh"],
        Environment: ["SOURCE=cpi.c"],
```

Для передачи файлов с локальной машины клиента на удаленный UNICORE-сайт локальные и удаленные имена файлов прописываются в секции «Imports». Следует заметить, что не нужно указывать полный путь файла на сервере, а только его имя. Он будет размещен в рабочий каталог текущего задания. Например:

```
{
 Imports: [
  { File: cpi.c, To: cpi.c },
  { File: script.sh, To: script.sh },

]
}
```

Для получения файлов после выполнения заданий удаленное и локальное имя файла указываются в секции «Exports». Локальное имя файла может быть иметь абсолютный или относительный путь. В последнем случае он будет помещен в каталог получения результатов по умолчанию. Например:

```
{
   Exports: [
    { File: "pi.txt", To: "pi.txt" },
   ]
}
```

Для передачи файлов между удаленными ресурсами используются секции «Stage in» и/или «Stage out» (соответственно до и после запуска заданий). Например:

```
  Stage in: [
        {
          From: "u6://DEMO-SITE/Home/examples/cpi.c",
          To: "cpi.c"
        }
  ],
    Stage out: [
        {
          From: stdout,
          To: "u6://DEMO-SITE/Home/output/stdout"
        }
  ],
```

Чтобы приложение читало стандартный ввод из файла, необходимо в файле задания указать параметр «Stdin». Например:

```
Stdin: filename,
```

Чтобы запустить задание на конкретном грид-сайте необходимо указать имя сайта. Например:

```
Site: "DEMO-SITE",
```

Для указания ресурсов, требуемых для выполнения задания, требуется ввести параметры в секцию «Resources». Ни один из них не является обязательным. Например:

```
  Resources: {

    #выделение памяти (в байтах)
    Memory: 268435456 ,

    #время на выполнение (в процессоросекундах)
    Runtime: 86400 ,

    #количество узлов
    Nodes: 2 ,

    #количество процессоров на узел
    CPUsPerNode: 8 ,
}
```

В случае, если сертификат пользователя позволяет запускать задания под различными пользователями (xlogin), необходимо дополнительно указать параметр «User name». Например:

```
User name: loginname,
```

Чтобы указать e-mail, на который системе пакетной обработки заданий следует высылать сообщения о завершении заданий, необходимо задать параметр «`User email`». Например:

```
User email: user@mail.by,
```

Для указания имени задания необходимо задать параметр «`Name`». Например:

```
Name: Test job,
```

3.1.3 Запуск заданий

а) Перейти в рабочий каталог, содержащий файлы тестового задания.

б) После выполнения настройки клиента командной строки UCC (см. п.2.4 данного руководства) необходимо выполнить команду соединения с реестром UNICORE, URL которого указан в файле `preference`:

```
ucc connect
```

При выполнении команды соединения с реестром UNICORE URL реестра можно указать в командной строке с помощью ключа «`-r`», например,

```
ucc connect -r https://grid1.basnet.by:8080/SKIF_GRID/services/Registry?res=default_registry
```

в) Просмотреть список UNICORE-сайтов с помощью команды:

```
ucc list-sites
```

Клиент командной строки UCC поддерживает запуск заданий в синхронном и асинхронном режимах. В первом случае после запуска задания клиент остается подключенным к грид-сайту в течение всего времени выполнения задания. После завершения задания клиент автоматически получает файлы с результатами выполнения задания (см. п 3.1.3.1).

В асинхронном режиме клиент подключается к грид-сайту, отправляет задание на выполнение, при этом, в текущую рабочую директорию записывается файл `job` с описанием задания. Клиент может опрашивать состояние выполнения задания и после получения статуса успешного завершения задания получить результаты выполнения задания с UNICORE-сайта на локальную машину с помощью команды получения результатов (см. п. 3.1.3.2).

3.1.3.1 Запуск заданий в синхронном режиме

Для запуска задания в синхронном режиме необходимо выполнить в командной строке:

```
ucc run -s <SITENAME> <файл задания>
```

Например:

```
ucc run -s DEMO-SITE test.u
SUCCESSFUL exit (0)
/opt/ucc/samples/cpi/./9a7952ef-d952-4f80-9617-0f9b50b34b33.stdout
/opt/ucc/samples/cpi/./9a7952ef-d952-4f80-9617-0f9b50b34b33.stderr
/opt/ucc/samples/cpi/./9a7952ef-d952-4f80-9617-0f9b50b34b33.pi.txt
/opt/ucc/samples/cpi/./9a7952ef-d952-4f80-9617-0f9b50b34b33.properties
```

В случае успешного выполнения задания (сообщение:SUCCESSFUL exit code: 0) в текущий рабочий каталог локальной машины клиента с UNICORE-сайта будут скопированы выходные файлы `9a7952ef-d952-4f80-9617-0f9b50b34b33.stdout`, `9a7952ef-d952-4f80-9617-0f9b50b34b33.stderr` и `9a7952ef-d952-4f80-9617-0f9b50b34b33.pi.txt`.

3.1.3.2 Запуск заданий в асинхронном режиме

Для запуска задания в асинхронном режиме необходимо добавить ключ «`-a`» в команде запуска задания:

```
ucc run -s <DEMO-SITE> -a <файл задания>
```

Например:

```
ucc run -s DEMO-SITE -a test.u
```

```
/home/user/cpi/./831789d9-c782-4356-b76a-2d8d59d9ee12.job
/home/user/cpi/./831789d9-c782-4356-b76a-2d8d59d9ee12.properties
```

При корректном выполнении в текущей директории должен появиться файл <job-id>.job, в котором содержится идентификатор задания, адрес сайта, на котором оно запущено, и т.д.

Этот файл в дальнейшем будет использован для любых операций с заданием. Чтобы получить статус задания, необходимо выполнить команду:

```
ucc get-status ./<job-id>.job
```
где <job-id> – идентификатор задания.

Например:
```
ucc get-status ./831789d9-c782-4356-b76a-2d8d59d9ee12.job
SUCCESSFUL exit code: 0
```

После получения сообщения об успешном выполнении задания (SUCCESSFUL exit (0)) для получения результатов выполнения задания с UNICORE-сайта на локальную машину следует выполнить команду:

```
ucc get-output ./<job-id>.job
```
где <job-id> – идентификатор задания.

Например:
```
ucc get-output ./831789d9-c782-4356-b76a-2d8d59d9ee12.job
SUCCESSFUL exit code: 0
/home/user/cpi/./831789d9-c782-4356-b76a-2d8d59d9ee12.stdout
/home/user/cpi/./831789d9-c782-4356-b76a-2d8d59d9ee12.stderr
/home/user/cpi/./831789d9-c782-4356-b76a-2d8d59d9ee12.pi.txt
```

3.1.3.3 Запуск заданий с указанием URL реестра UNICORE-сайта

Для запуска задания с указанием URL реестра конкретного UNICORE-сайта (без обращения к реестру национальной грид-сети), необходимо использовать ключ *«-r»* в команде запуска задания:

```
ucc run -r <URL> <файл задания>
```

Например:
```
ucc run -r https://gpu1.basnet.by:8080/SKIF_GPU/services/Registry?res=default_registry -v test.u
```

3.1.4 Передача данных

С помощью клиента командной строки UCC можно передавать данные на сервер, загружать данные с него, а также инициировать передачи данных типа «сервер-сервер». Удаленные ресурсы могут быть указаны двумя способами. Первый способ – это указание полного адреса (URI), который включает протокол, сервер и путь к файлу, например:

```
RBYTEIO:https://server:8080/DEMO-
```

```
SITE/services/StorageManagement?res=default_storage#/file.txt
```

Второй способ – более простой, он требует указания только имени сайта и названия хранилища данных. В приведенном ниже примере – это «DEMO-SITE» и «Home» соответственно. Можно указывать «u6» вместо «unicore6» в URL удаленных хранилищ.

```
unicore6://DEMO-SITE/Home/file.txt
```

или

```
u6://DEMO-SITE/Home/file.txt
```

Также можно обращаться непосредственно в рабочие каталоги заданий по их идентификатору. Список идентификаторов заданий можно получить, используя команду

```
ucc list-jobs
```

Например:

```
unicore6://DEMO-SITE/1f3bc2e2-d814-406e-811d-e533f8f7a93b/outfile
```

Для передачи файлов в клиенте командной строки UCC служат следующие операции:

`get-file` – для загрузки файлов с удаленного хранилища.
Например:

```
ucc get-file -s u6://DEMO-SITE/Home/file.txt -t file.txt
```

`put-file` – для отправки файлов на удаленное хранилище.
Например:

```
ucc put-file -s cpi.c -t u6://DEMO-SITE/Home/cpi.c
```

`copy-file` – для передачи файла с сервера на сервер.
Например:

```
ucc copy-file -s u6://DEMO-SITE-1/Home/file1 -t u6://DEMO-SITE-
2/Home/file2
```

`resolve` – для получения «реального» адреса удаленного ресурса «unicore6://».
Например:

```
ucc resolve u6://DEMO-SITE/Home
https://server:8080/DEMO-SITE/services/StorageManagement?res=7d077b00-a8a0-410e-bbeb-0ef318ce181f
```

3.2 Выполнение программы UNICORE Rich Client

3.2.1 Запуск тестового задания

Для запуска тестового задания необходимо в главном окне приложения на вкладке «Grid Browser» выделить целевую систему для выполнения задания (рисунок 7) и в

контекстном меню, вызываемом с помощью правой кнопки мыши, выбрать пункт «create job» (рисунок 8).

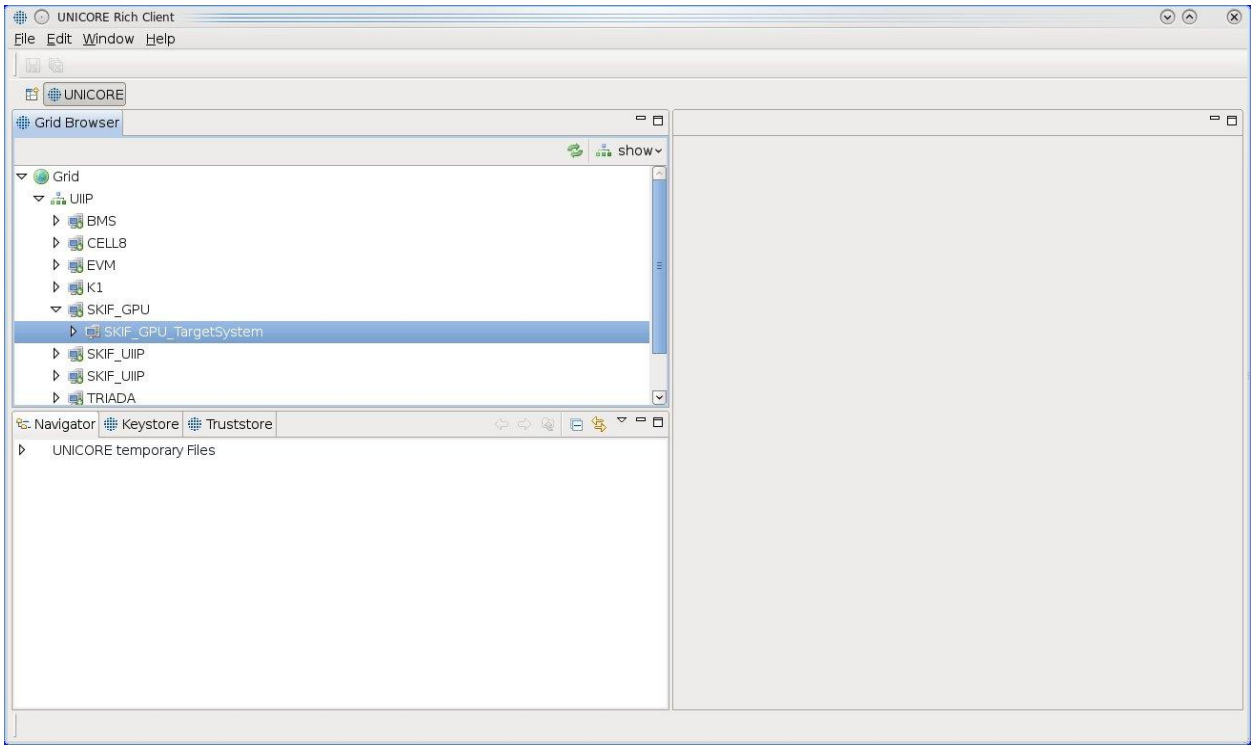

Рисунок 7 – Выбор целевой системы.

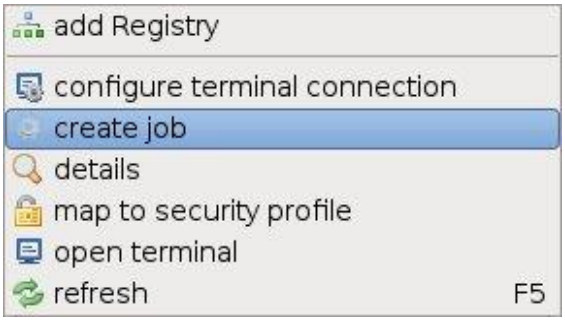

Рисунок 8 – Контекстное меню целевой системы.

Выбрать из списка типов заданий тип Script v.2.0 (рисунок.9).

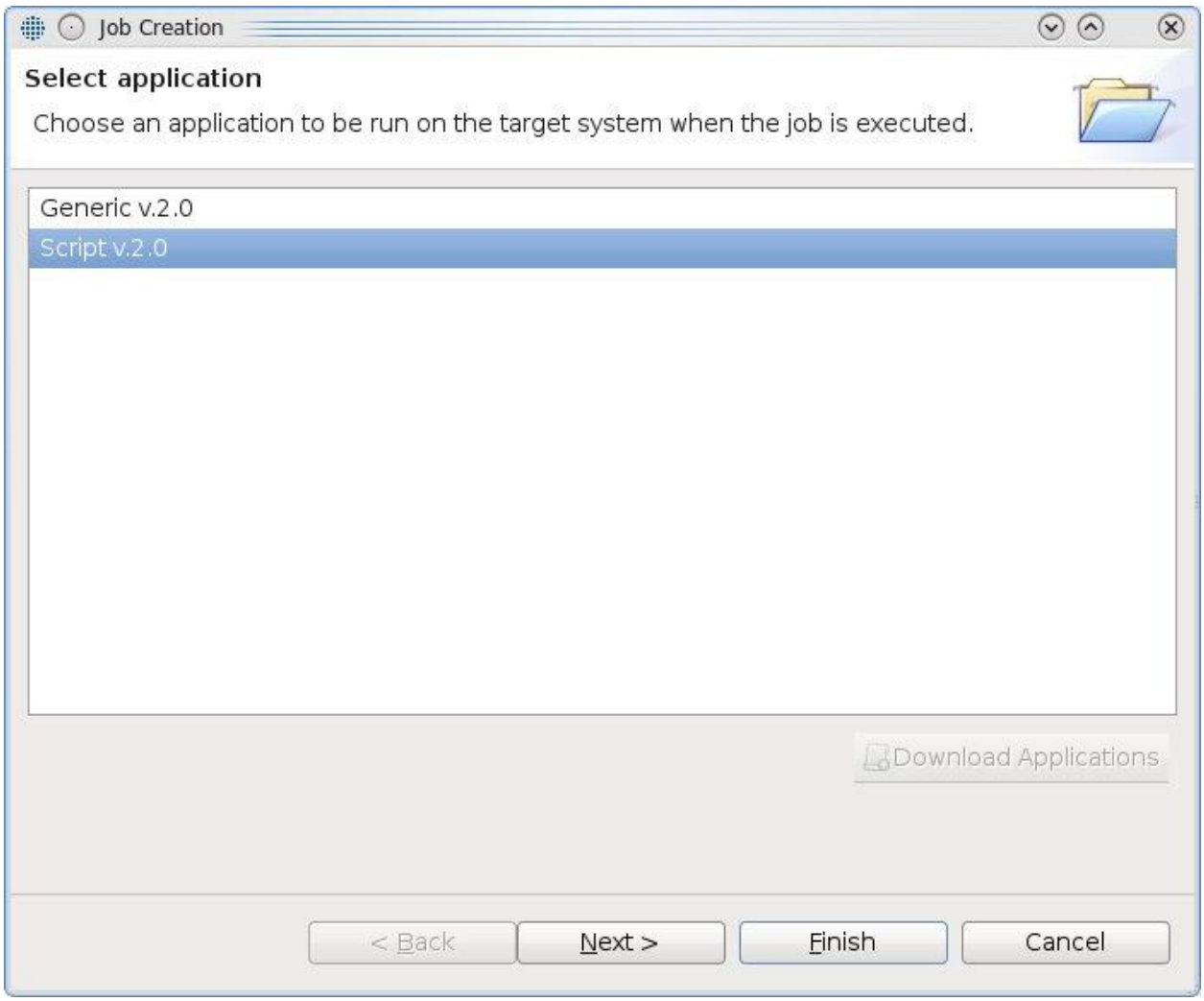

Рисунок 9 - Выбор типа задания.

В качестве тестового примера можно запустить MPI-программу вычисления числа Пи.

В окне определения задания на вкладке Script Input в поле «Name» ввести имя задания, например Test_cpi, в списке Application выбрать из списка приложений приложение Bash shell (рисунок 10).

В качестве приложения выбирается Bash shell с аргументом. В данном случае аргументом будет значение «script.sh» (из каталога `/opt/ucc/samples/cpi`). С помощью пункта меню File - > Open открыть файл `script.sh`.

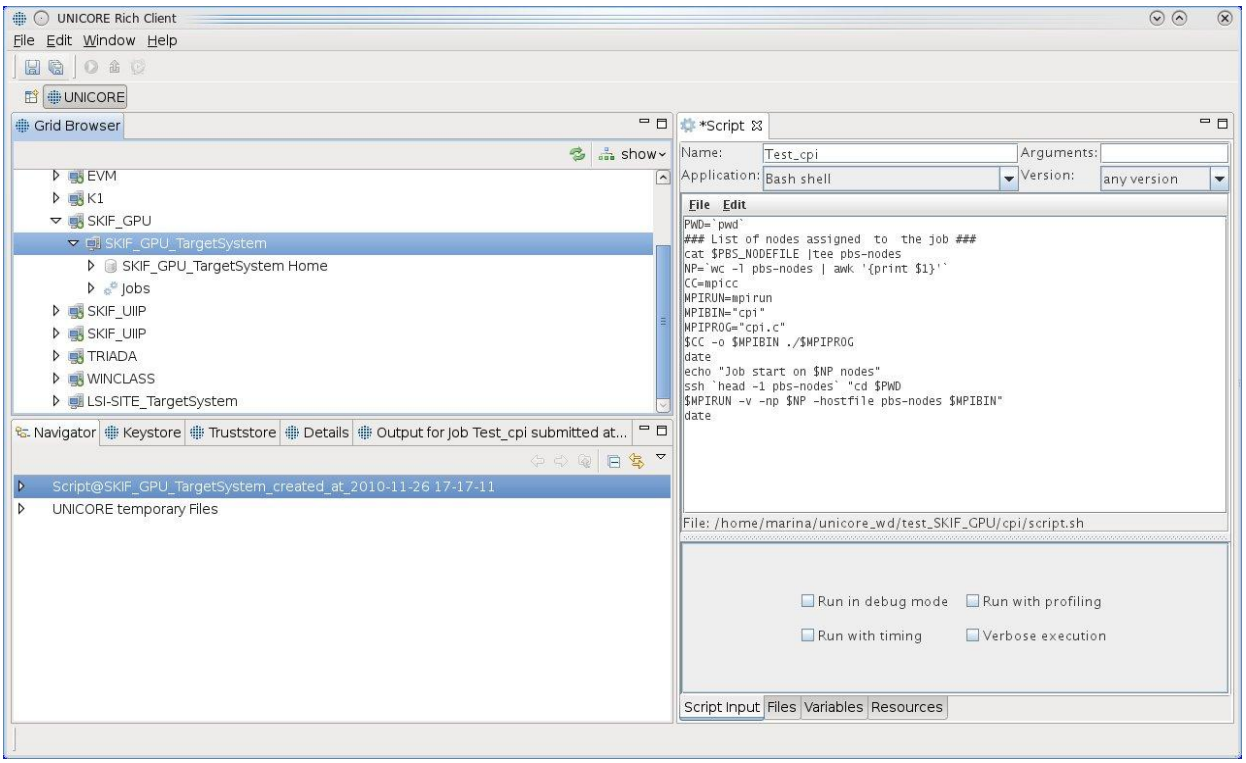

Рисунок 10 – Вкладка Script Input окна определения задания.

На вкладке Files окна определения задания необходимо указать передаваемые на грид-сайт и получаемые с грид-сайта файлы (рисунок 11). Эта операция производится путем вызова контекстного меню и выбора пункта меню «add File» в соответствующем поле.

На грид-сайт необходимо передать файлы `cpi.c` и `script.sh` из каталога `/opt/ucc/samples/cpi/` и после окончания задания на локальный компьютер должны быть переданы с грид-сайта выходные файлы `stderr`, `stdout` и `pi.txt`.

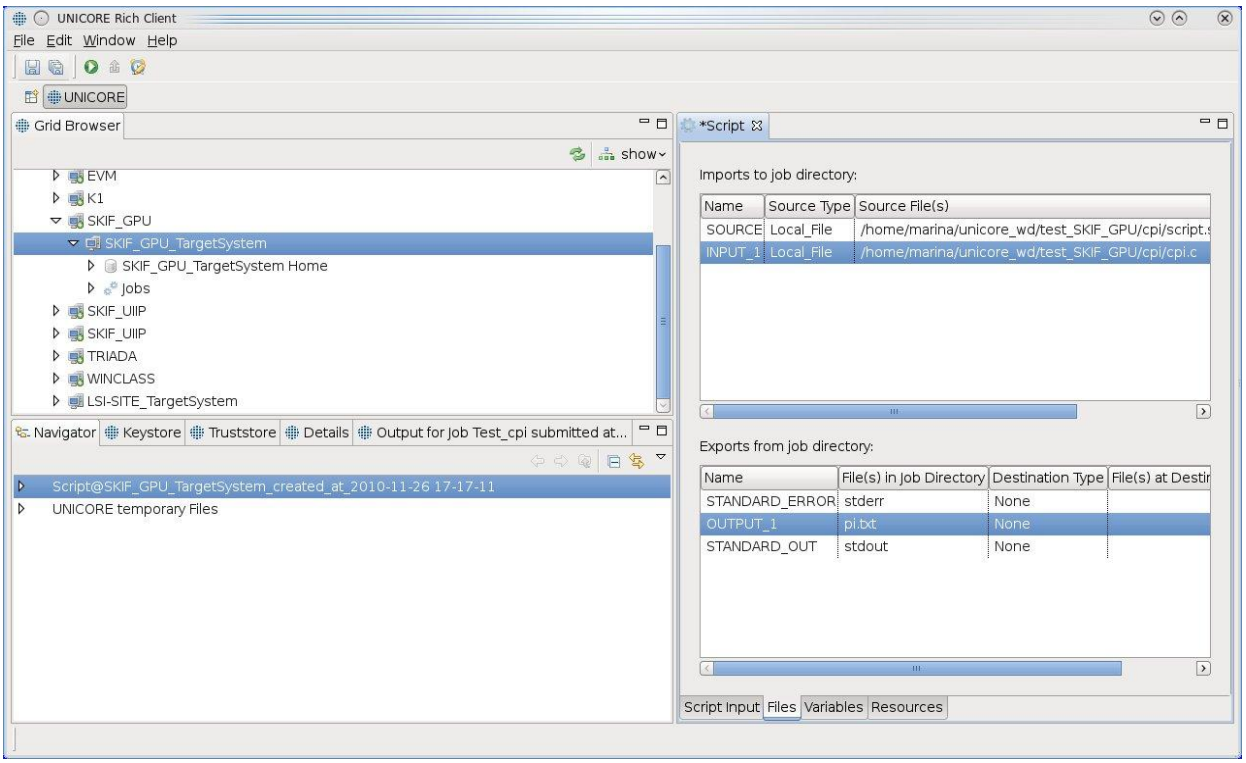

Рисунок 11 - Определение файлов, передаваемых с локальной машины на грид-сайт и обратно.

На вкладке Resources окна определения задания необходимо указать необходимые для выполнения задания ресурсы (рисунок 12):

- число узлов;
- число процессов на узле;
- объем требуемой памяти;
- время выполнения задания.

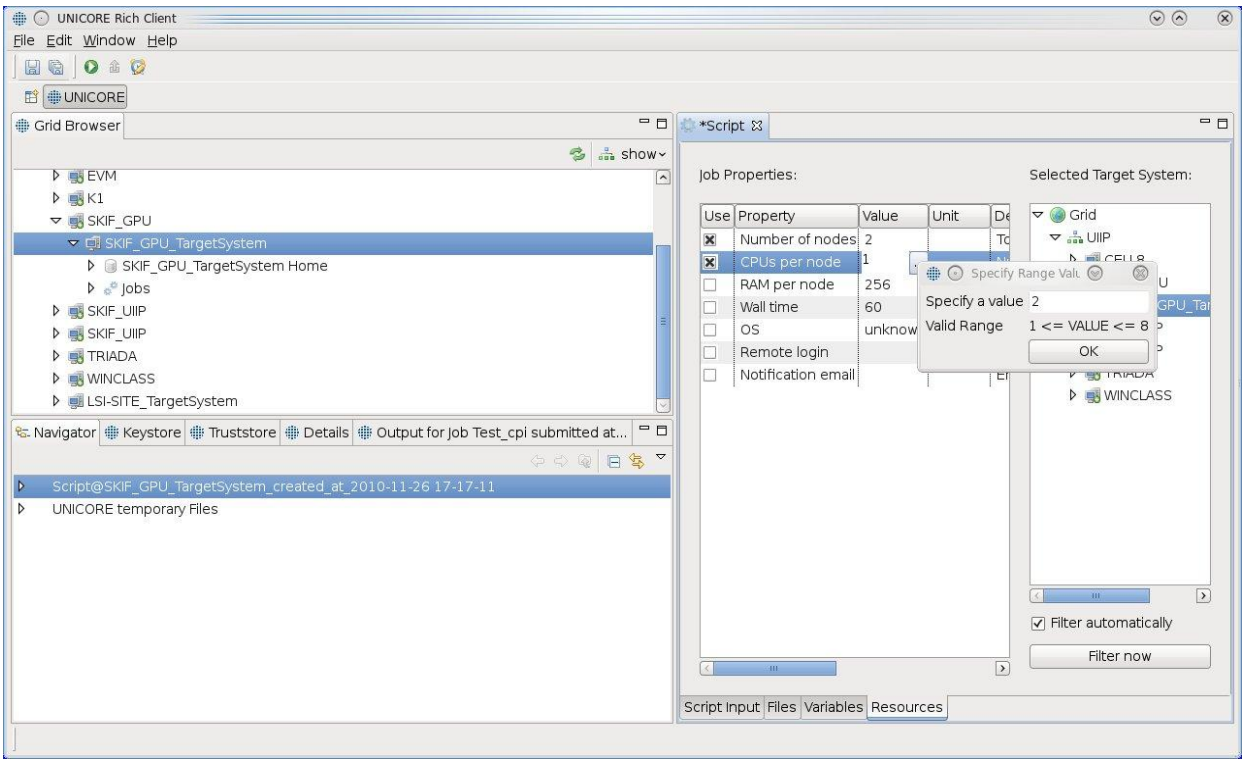

Рисунок 12 – Определение ресурсов задания.

Для отправки задания на выполнение необходимо нажать кнопку «submit»  .

Мониторинг выполнения задания осуществляется путем вызова контекстного меню на имени задания в дереве «Grid» и выбора пункта меню «refresh». После получения статуса «SUCCESSFUL» (рисунок 13) результаты задания можно получить с грид-сайта на локальный компьютер, вызвав пункт «fetch output files» в контекстном меню (рисунок 14).

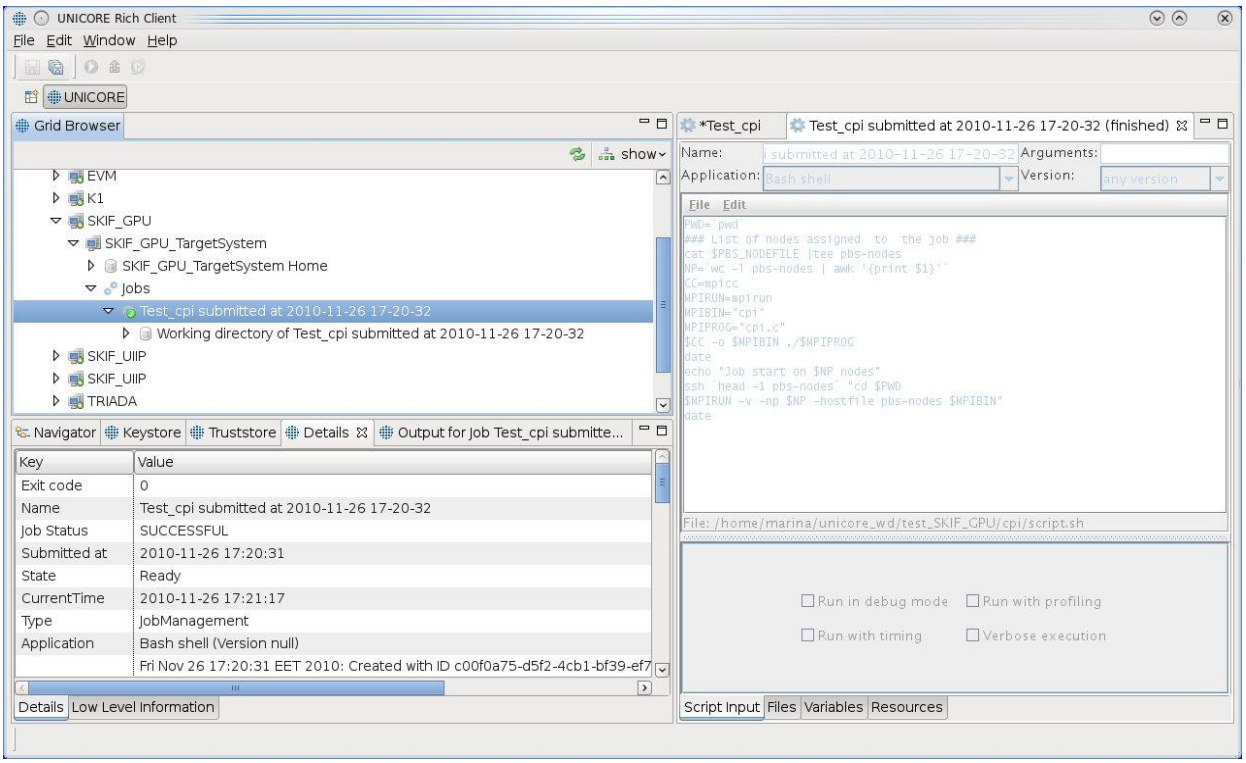

Рисунок 13 – Успешное завершение тестового задания.

Рисунок 14 – Контекстное меню задания.

После получения результатов выполнения задания на локальный компьютер, просмотреть выходные файлы на вкладке Output for job Test_cpi (рисунок 15).

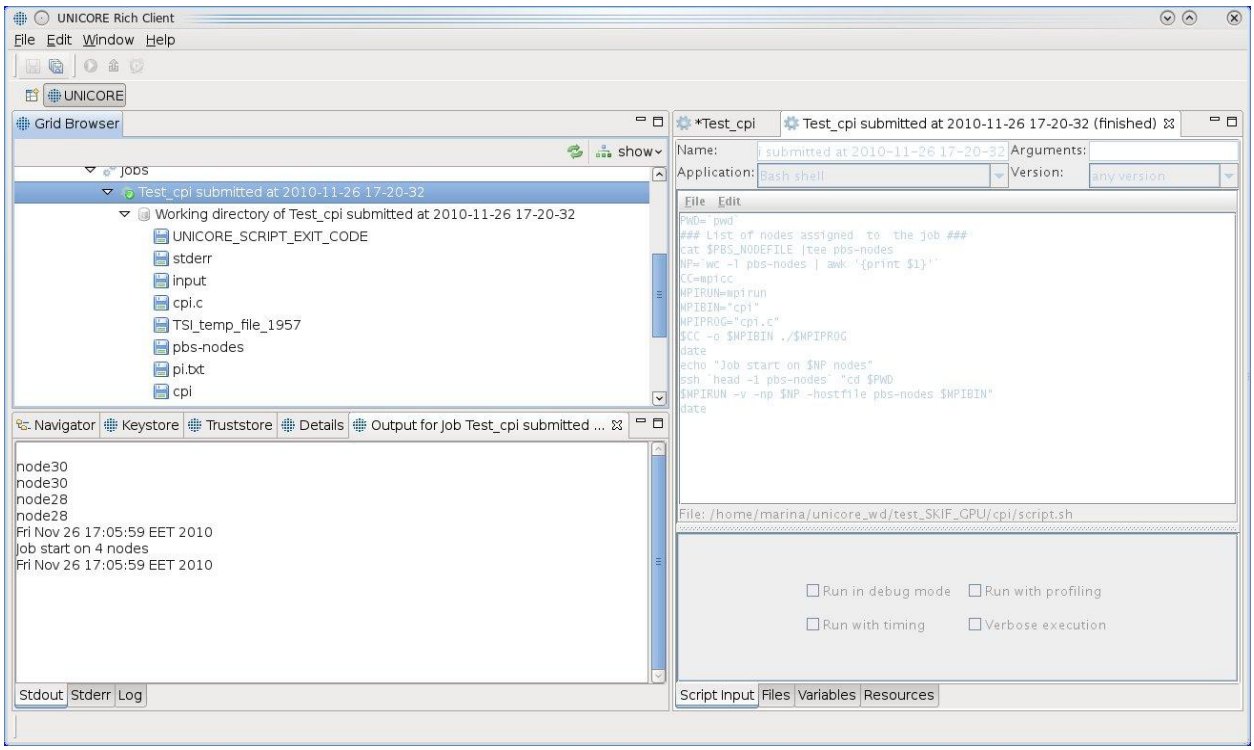

Рисунок 15 – Результаты выполнения задания.

3.3 Выполнение программы UFM

Информацию по выполнению файлового менеджера UNICORE (UFM) можно получить в руководстве оператора файлового менеджера.

3.4 Установка прикладного программного обеспечения на UNICORE-сайт

При работе на UNICORE-сайте каждому грид-пользователю автоматически выделяется учетная запись вида aguXX (XX – некоторое число) из пула анонимных пользователей.

Установка грид-пользователем прикладного программного обеспечения на UNICORE-сайт выполняется в разделяемый каталог `/share/unicore/data` хранилища грид-сайта с помощью выполнения скрипта грид-задания вида:

```
id
cd $HOME
wget http://somewhere/package.tar.gz
tar xfz package.tar.gz
cd package-0.0.1
```

```
./configure --prefix=/share/unicore/data/package
make -j8
make install
```

В приведенном выше примере пакет с условным названием `package` будет установлен в каталог `/share/unicore/data/package` и будет доступен для работы с любого вычислительного узла UNICORE-сайта. Для редактирования файлов в каталоге `/home/aguXX/package-0.0.1` можно использовать файловый менеджер UNICORE (UFM).

В заказе ресурсов для грид-задания, выполняющего сборку и установку пакетов достаточно заказать только один узел. Например, для выполнения скрипта из приведенного выше примера необходимо заказать один узел и 8 процессов на этом узле.

Определить переменные среды, необходимые для запуска задач с исполняемыми файлами пакета, можно с помощью следующих команд в скрипте грид-задания:

```
export PATH=/share/unicore/data/package/bin:$PATH
export LD_LIBRARY_PATH=/share/unicore/data/package/lib
```

Для выполнения грид-заданий на UNICORE-сайтах необходимо корректно заказывать требуемое количество узлов и процессов на узле с учетом технических характеристик узлов грид-сайтов. Примеры грид-заданий для UNICORE-сайтов национальной грид-сети приведены на сайте Операционного центра национальной грид-сети http://noc.grid.basnet.by.

4 СООБЩЕНИЯ ОПЕРАТОРУ

4.1 Получение справки об использовании UCC

Для получения справки необходимо запустить ucc без параметров.

```
ucc

UCC 1.3.1
Usage: ucc <command> [OPTIONS] <args>
The following commands are available:
Data management:
 ls                   - list a storage
 rm                   - Remove a remote file or directory
 copy-file-status     - check status of a copy-file
 get-file             - get remote files
 find                 - find files on storages
 resolve              - resolve remote location
 mkdir                - create a directory remotely
 copy-file            - copy remote files
 put-file             - puts a local file to a remote server
General:
 create-storage       - create a storage service instance
 connect              - connect to UNICORE
 list-storages        - list the available remote storages
 list-applications    - lists applications on target systems
 list-jobs            - list your jobs
 list-sites           - list remote sites
 system-info          - Checks the availability of services.
Job execution:
 run                  - run a job through UNICORE 6
 get-status           - get job status
 abort-job            - abort a job
 batch                - run ucc on a set of files
 get-output           - get output files
OGSA-BES:
 bes-list-att         - provides information about the UNICORE 6 BES
Interface
 bes-terminate-job    - terminate bes activity
 bes-submit-job       - run a job through UNICORE 6 BES Interface
 bes-list-jobs        - list jobs running on BES.
 bes-job-status       - get bes activity status
Other:
 shell                - Starts an interactive UCC session
 issue-delegation     - Allows to issue a trust delegation assertion
 connect-to-testgrid  - Get credentials for the public testgrid
 wsrf                 - perform a WSRF operation
 cip-query            - query a CIS Infoprovider at a UNICORE site
 plan                 - Plan job
 run-groovy           - run a Groovy script
Workflow:
 workflow-trace       - trace info on a workflow in Chemomentum
 workflow-control     - offers workflow control functions
 workflow-submit      - submit a workflow
 workflow-info        - lists info on workflows.
 broker-run           - submit a work assignment to a service orchestrator
Enter 'ucc <command> -h' for help on a particular command.
```

4.2 Получение информации о ресурсах UNICORE-сайтов

а) Выполнить команду соединения с реестром UNICORE, URL которого указан в файле `preference`

Например,

```
ucc connect
You can access 1 target system(s).
```

б) Для получения списка доступных UNICORE-сайтов необходимо выполнить команду «`ucc list-sites`»:

Например,

```
ucc list-sites
DEMO-SITE https://server:8080/DEMO-SITE/services/TargetSystemService?res=740c7ae7-2241-4cc5-838e-628804e45435
```

в) Для получения списка установленных на UNICORE-сайте приложений необходимо выполнить команду «`ucc list-applications –s SITENAME`»,

где `SITENAME` - имя UNICORE-сайта

Например,

```
ucc list-applications –s DEMO-SITE
Applications on target system <DEMO-SITE>
Date 1.0
Bash shell 3.2.18
C shell 6.15
Perl 5.8.8
Python Script 2.5.1
Applications on target system <SKIF_UIIP>
Date 1.0
Bash shell 3.2.18
C shell 6.15
Perl 5.8.8
Python Script 2.5.1
```

г) Для просмотра списка идентификаторов заданий грид-пользователя на UNICORE-сайте выполнить команду:

```
ucc list-jobs -s SITENAME
```

Например,

```
ucc list-jobs -s DEMO-SITE
https://server:8080/DEMO-SITE/services/JobManagement?res=9f9a1df6-f967-4534-bf99-421f7425b718
```

4.3 Режим отладки

Каждая команда клиента командной строки UCC может быть запущена с дополнительным ключом «-v» (verbose), при указании которого программа будет писать в терминал все промежуточные действия, которые она выполняет. Например, при

подключении к UNICORE реестру, заданному в настройках файла preferences, с ключом «-v» будет выведена следующая информация:

```
ucc connect -v
[ucc connect] Verbose mode.
[ucc connect] Reading properties file </home/user/.ucc/preferences>
[ucc connect] Keystore = /home/user/.ucc/user-keystore.jks
[ucc connect] Keystore password given.
[ucc connect] Using default alias.
[ucc connect] Keystore type = jks
[ucc connect] Registry = https://server:8080/DEMO-SITE/services/Registry?res=default_registry
[ucc connect] Pinging registry.
[ucc connect] Registry PING OK, server reply: servertime=2009-12-02T17:56:49.560+02:00
[ucc connect] Output goes to <.>
[ucc connect] New TSSs will have a lifetime of <1> days.
[ucc connect] Connecting to https://server:8080/DEMO-
SITE/services/TargetSystemFactoryService?res=default_target_system_factory
You can access 1 target system(s).
```

При запуске тестового задания с ключом «-v» будет выведена следующая информация:

```
ucc run -v test.u
[ucc run] Verbose mode.
[ucc run] Reading properties file </home/user/.ucc/preferences>
[ucc run] Keystore = /home/user/.ucc/user-keystore.jks
[ucc run] Keystore password given.
[ucc run] Using default alias.
[ucc run] Keystore type = jks
[ucc run] Registry = https://server:8080/DEMO-SITE/services/Registry?res=default_registry
[ucc run] Pinging registry.
[ucc run] Registry PING OK, server reply: servertime=2009-12-02T17:57:06.080+02:00
[ucc run] Output goes to <.>
[ucc run] Synchronous processing = true
[ucc run] Adding job id to output file names = true
[ucc run] Assuming JSDL job file = false
[ucc run] Read job from <test.u>
[ucc run] Selected TSS at https://server:8080/DEMO-
SITE/services/TargetSystemService?res=740c7ae7-2241-4cc5-838e-628804e45435
[ucc run] Job submitted, job url=https://server:8080/DEMO-
SITE/services/JobManagement?res=8594e887-0c46-4ae6-b10a-e7b739b83e22
[ucc run] Importing USpace file 'cpi.c' from '/opt/ucc/samples/cpi/cpi.c'
[ucc run] Importing USpace file 'script.sh' from '/opt/ucc/samples/cpi/script.sh'
[ucc run] Job started.
QUEUED
SUCCESSFUL exit code: 0
[ucc run] Exporting USpace file 'stdout' to '/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-
e7b739b83e22.stdout'
/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-e7b739b83e22.stdout
[ucc run] Exporting USpace file 'stderr' to '/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-
e7b739b83e22.stderr'
/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-e7b739b83e22.stderr
[ucc run] Exporting USpace file 'pi.txt' to '/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-
e7b739b83e22.pi.txt'
/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-e7b739b83e22.pi.txt
/opt/ucc/samples/cpi/./8594e887-0c46-4ae6-b10a-e7b739b83e22.properties
```