

Робшкола – 2017

1. Соединитесь с Linux-кластером, используя Putty. Параметры доступа:

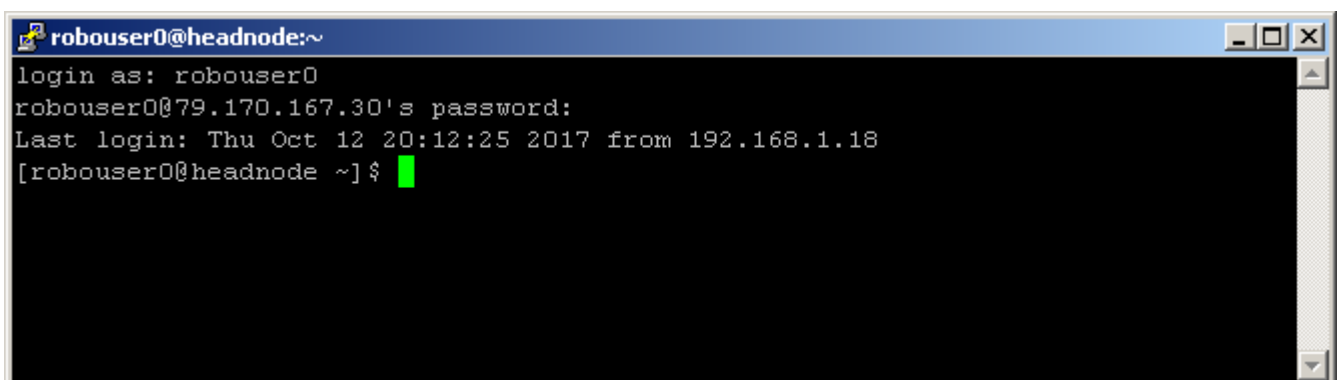
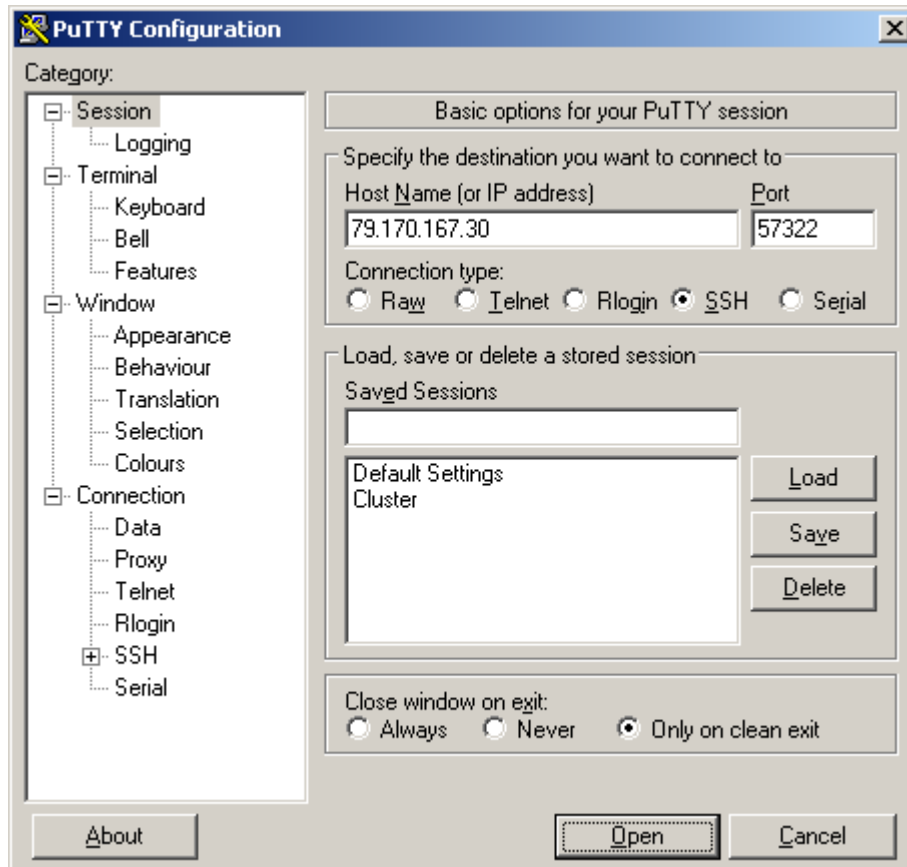
hostname: 79.170.167.30

port: 57322

username: robouser0, robouser1, ..., robouser9

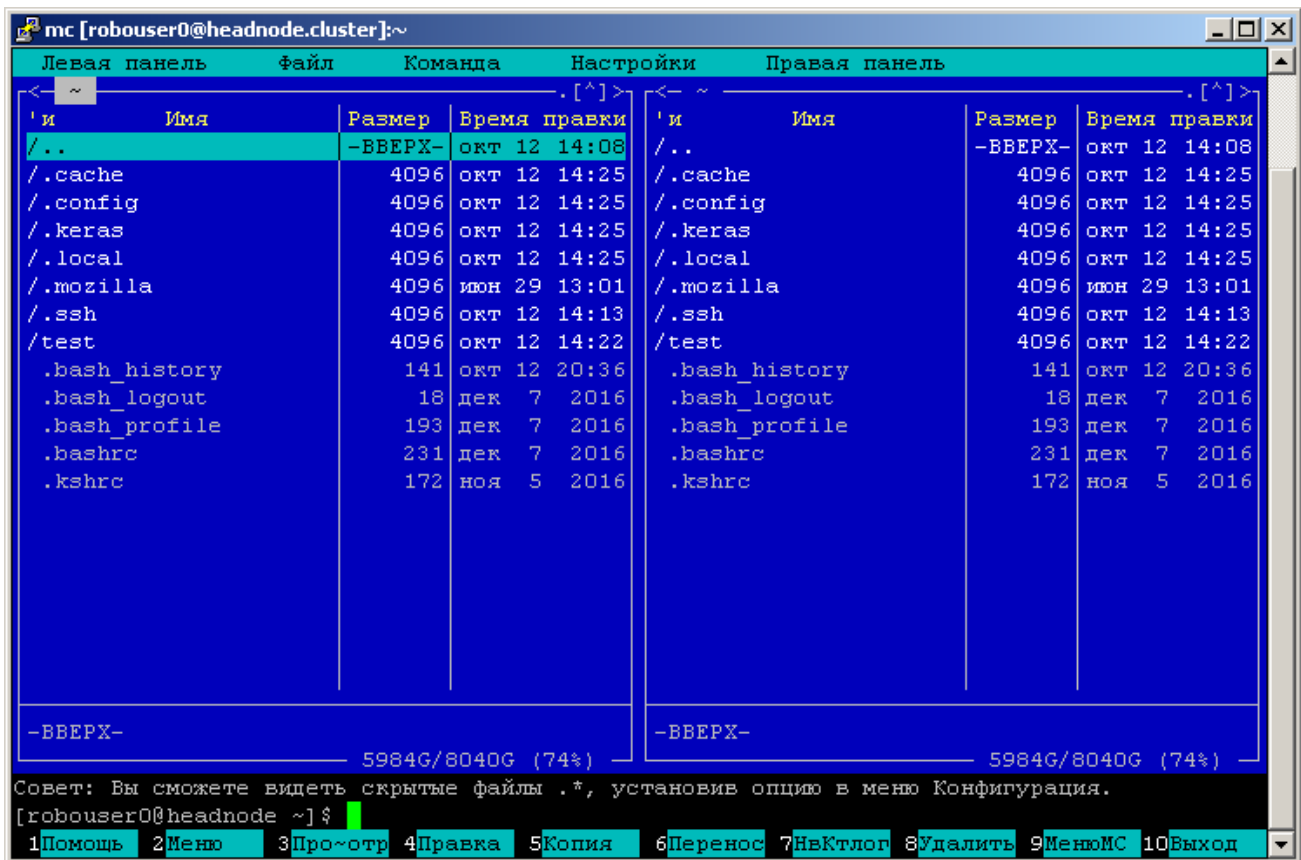
(имя пользователя выбирается в соответствии с номером рабочего компьютера)

password: RoboSchool2017

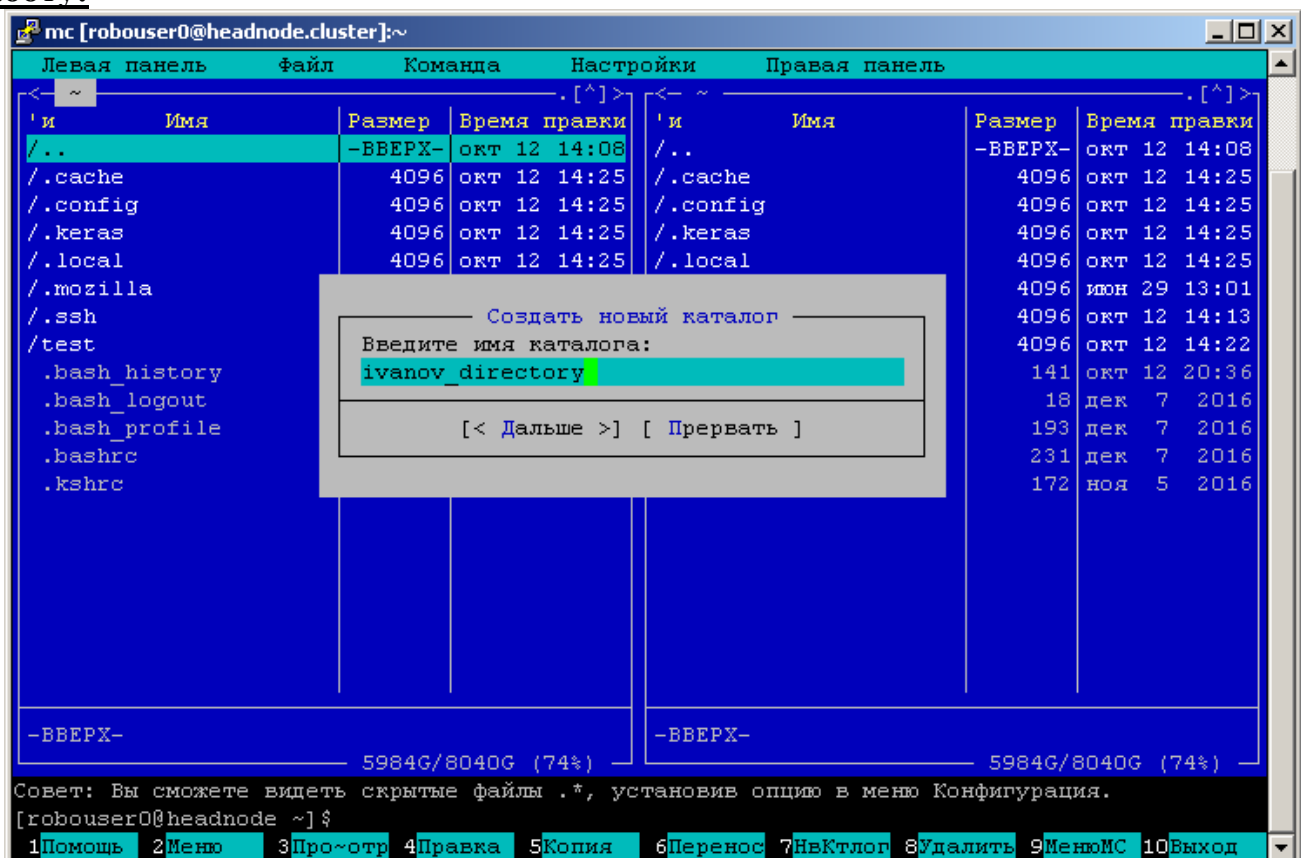


2. Для удобства дальнейшей работы запустите файловый менеджер Midnight Commander:

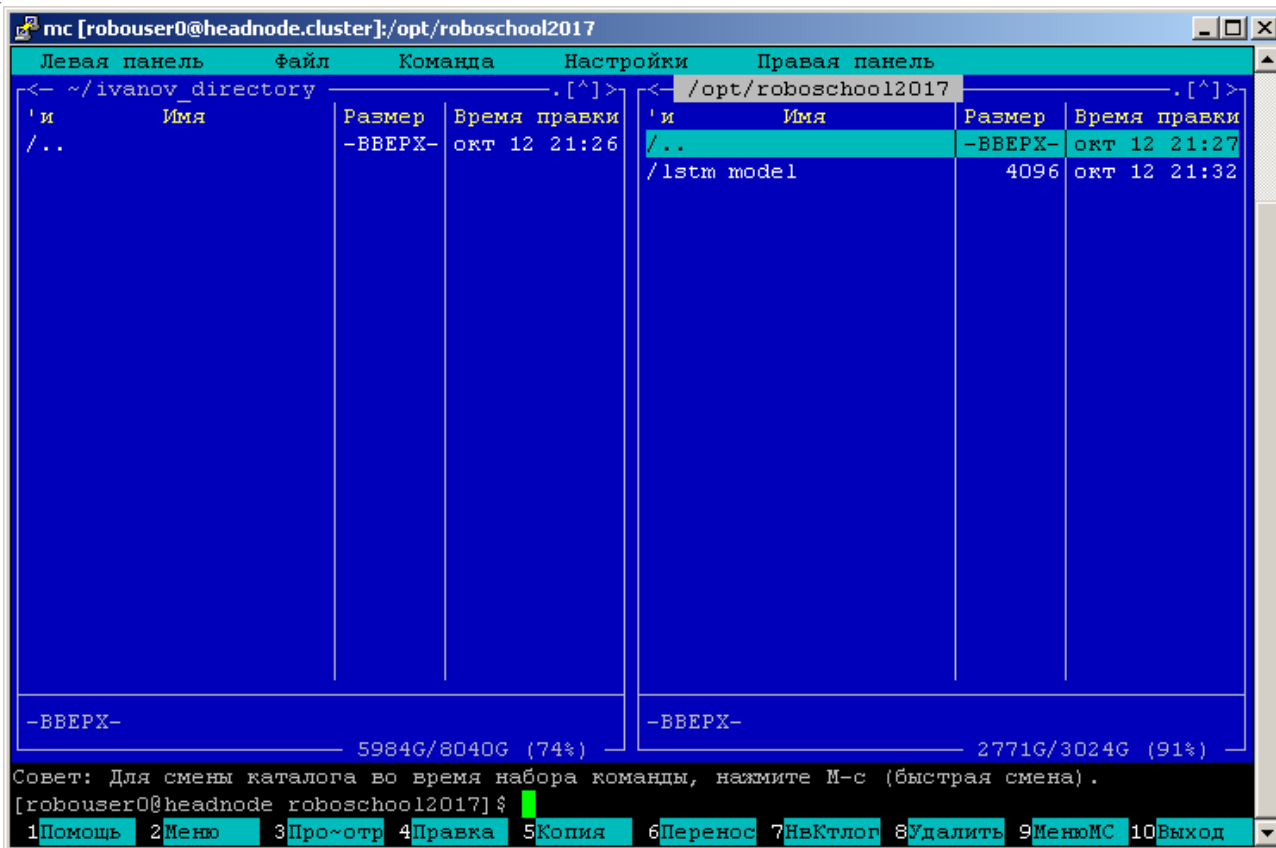
```
[robouser0@headnode ~]$ mc
```



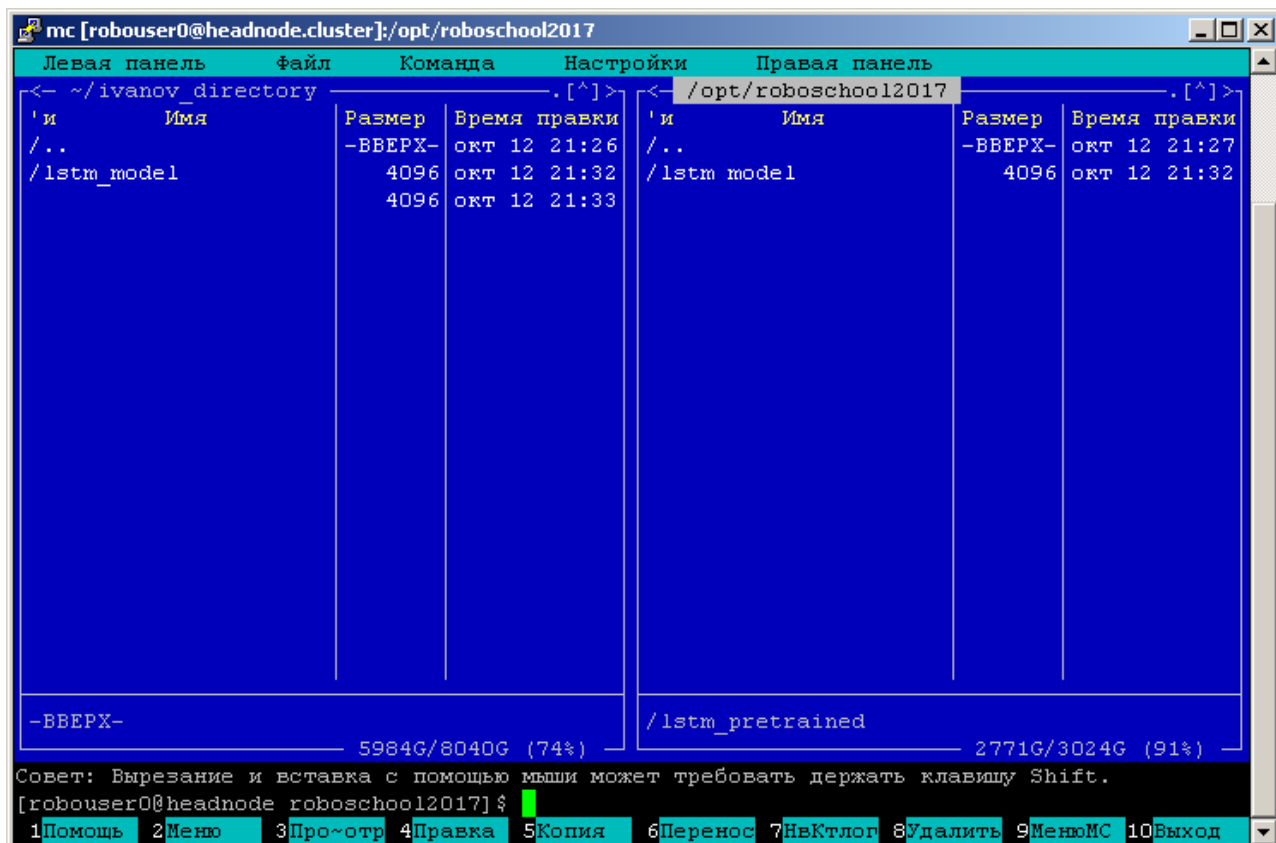
3. Файловый менеджер покажет содержимое домашней директории пользователя. Создайте в этой директории папку, в которой будут располагаться исследуемые файлы и программы, нажатием клавиши **F7**. **Директорию настоятельно рекомендуется называть по имени пользователей, выполняющих работу!**



4. Откройте в левом окне файлового менеджера только что созданную папку, а в правом — директорию /opt/roboschool2017. Между окнами менеджера можно переключаться клавишей Tab.



5. Скопируйте содержимое папки /opt/roboschool2017 в рабочую директорию с помощью клавиши F5.

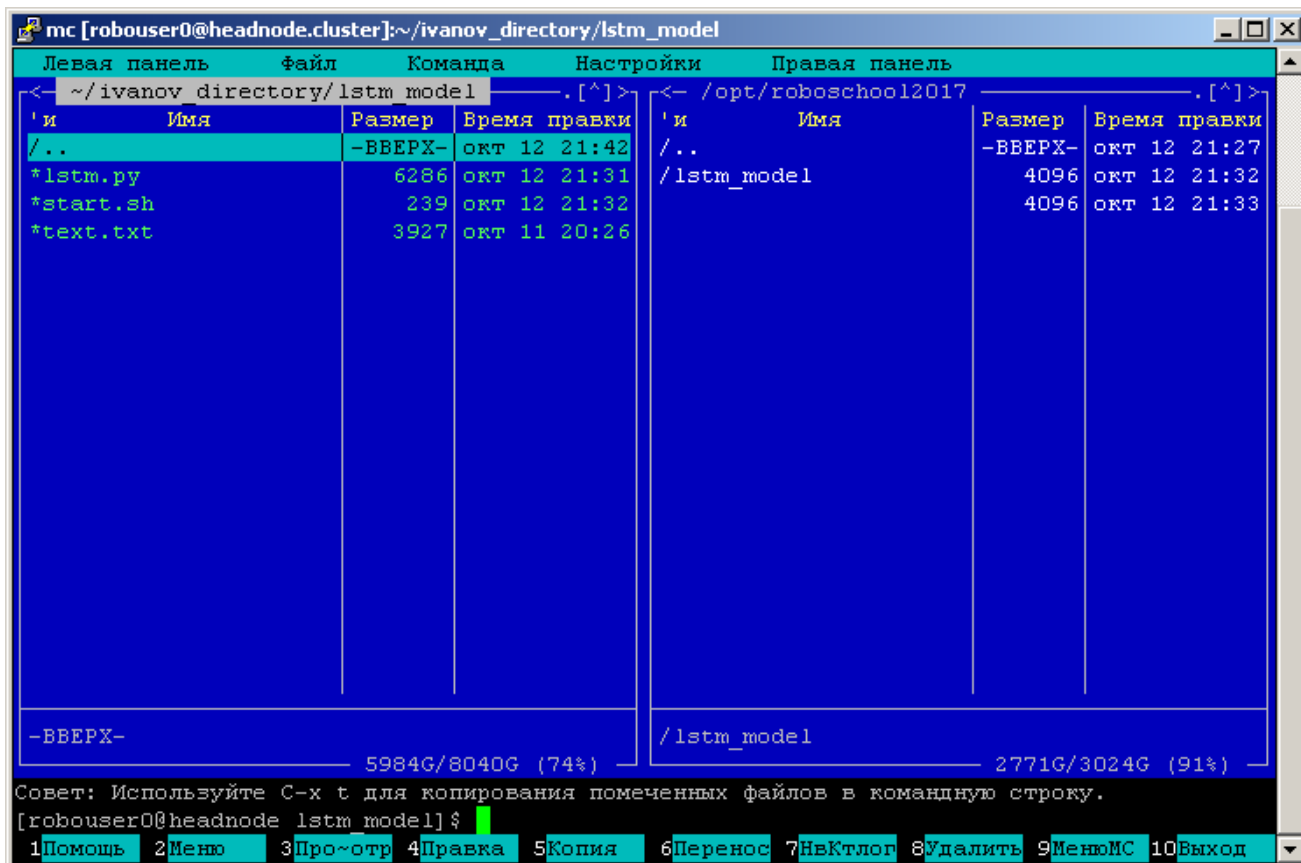


6. Откройте папку `lstm_model` в рабочей директории. В ней расположены три файла:

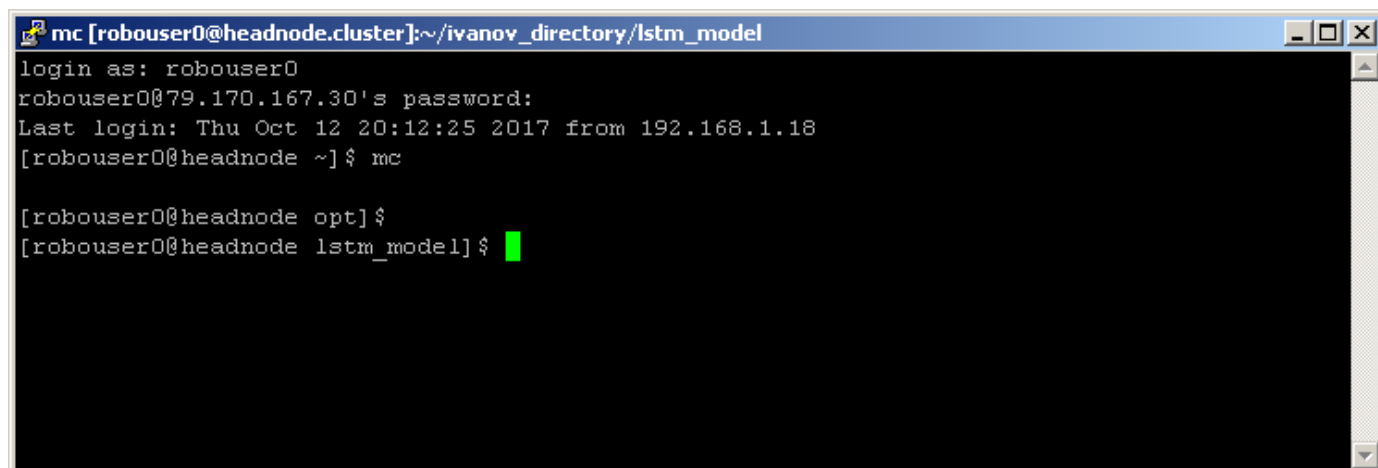
а) `lstm.py` — скрипт на языке Python, выполняющий обучение нейросети и получение предсказаний на его основе;

б) `start.sh` — скрипт запуска задачи обучения нейросети в системе очередей кластера Slurm;

в) `text.txt` — фрагмент произведения А.С. Пушкина «Евгений Онегин» (вообще говоря, текст может быть любым), на словах которого обучается нейросеть.

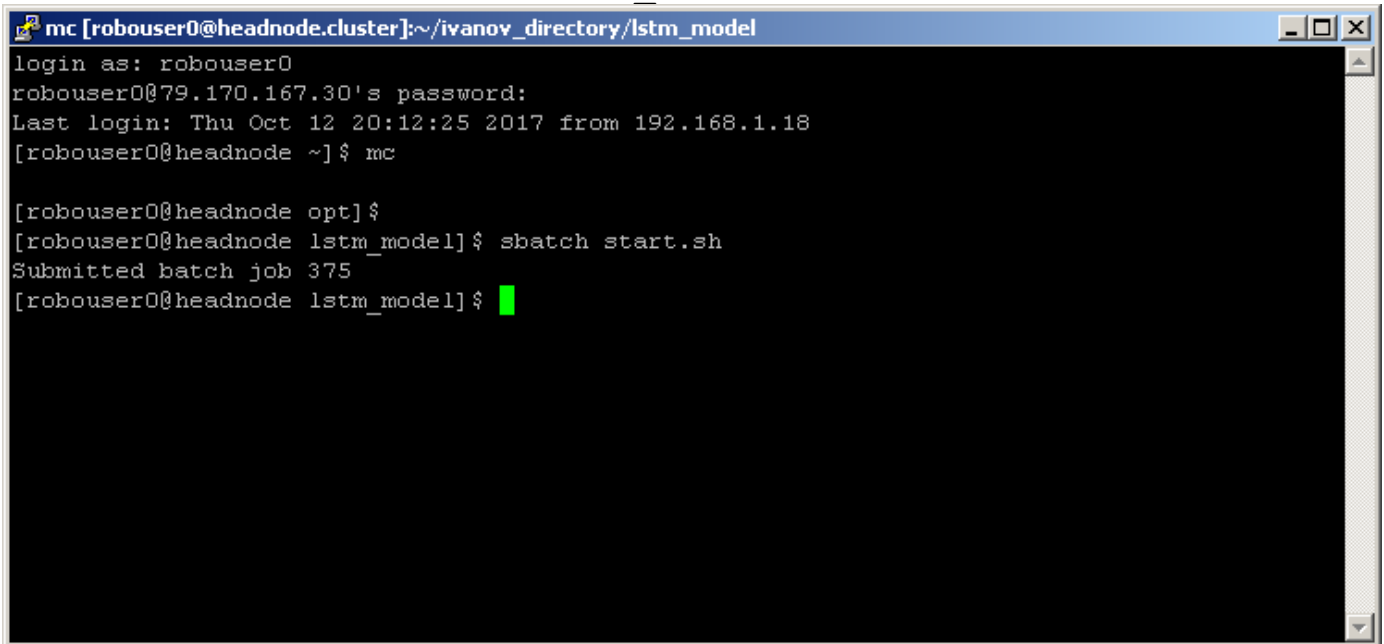


7. Переключите файловый менеджер в режим консольного ввода-вывода, нажав комбинацию клавиш `Ctrl-O` (если потребуется вернуться в оконный режим менеджера обратно, следует нажать `Ctrl-O` повторно).



8. Поставьте задачу обучения нейросети в очередь командой:

```
[robouser0@headnode lstm_model]$ sbatch start.sh
```

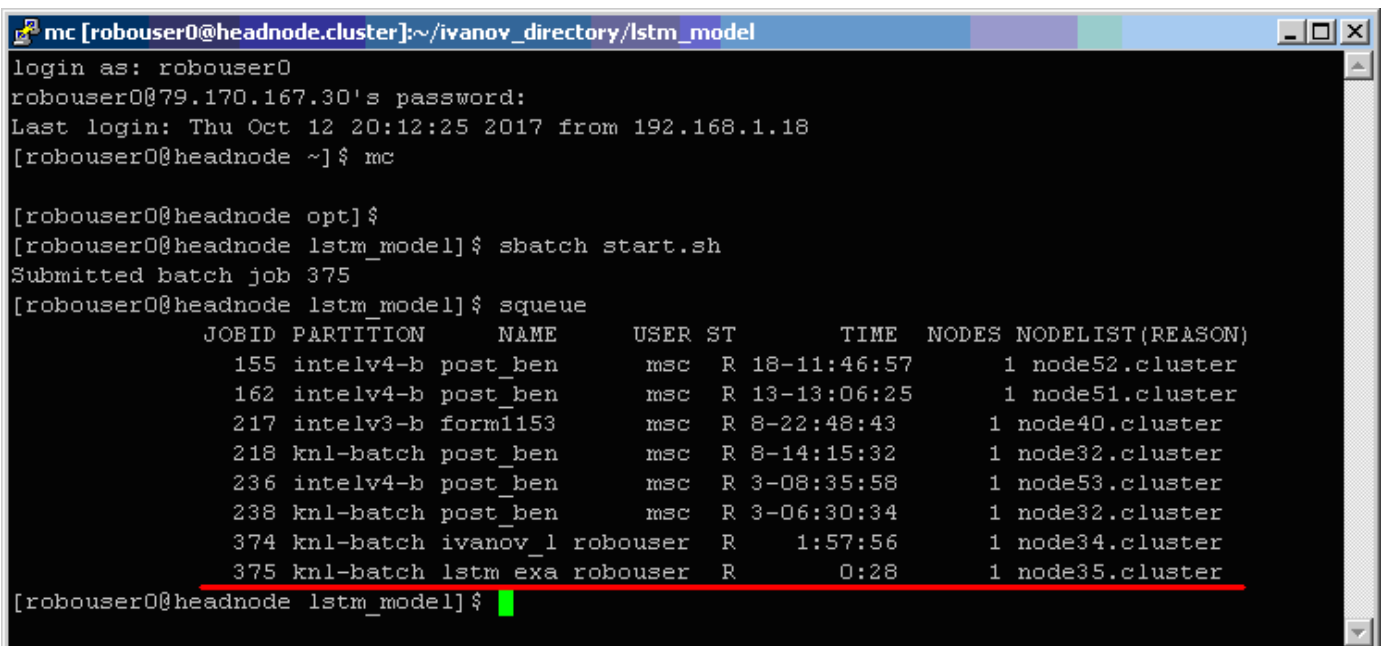


```
mc [robouser0@headnode.cluster]:~/ivanov_directory/lstm_model
login as: robouser0
robouser0@79.170.167.30's password:
Last login: Thu Oct 12 20:12:25 2017 from 192.168.1.18
[robouser0@headnode ~]$ mc

[robouser0@headnode opt]$
[robouser0@headnode lstm_model]$ sbatch start.sh
Submitted batch job 375
[robouser0@headnode lstm_model]$ █
```

9. После того, как задача поставлена в очередь, ее статус можно посмотреть командой:

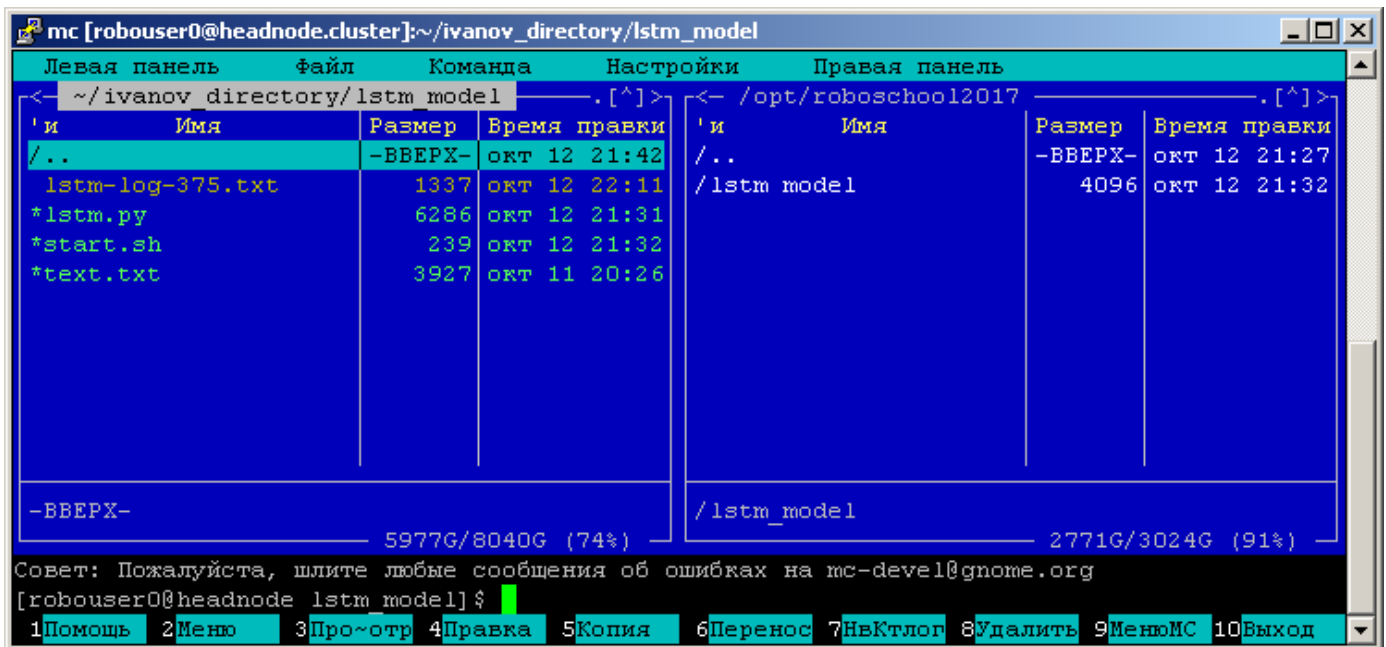
```
[robouser0@headnode lstm_model]$ squeue
```



```
mc [robouser0@headnode.cluster]:~/ivanov_directory/lstm_model
login as: robouser0
robouser0@79.170.167.30's password:
Last login: Thu Oct 12 20:12:25 2017 from 192.168.1.18
[robouser0@headnode ~]$ mc

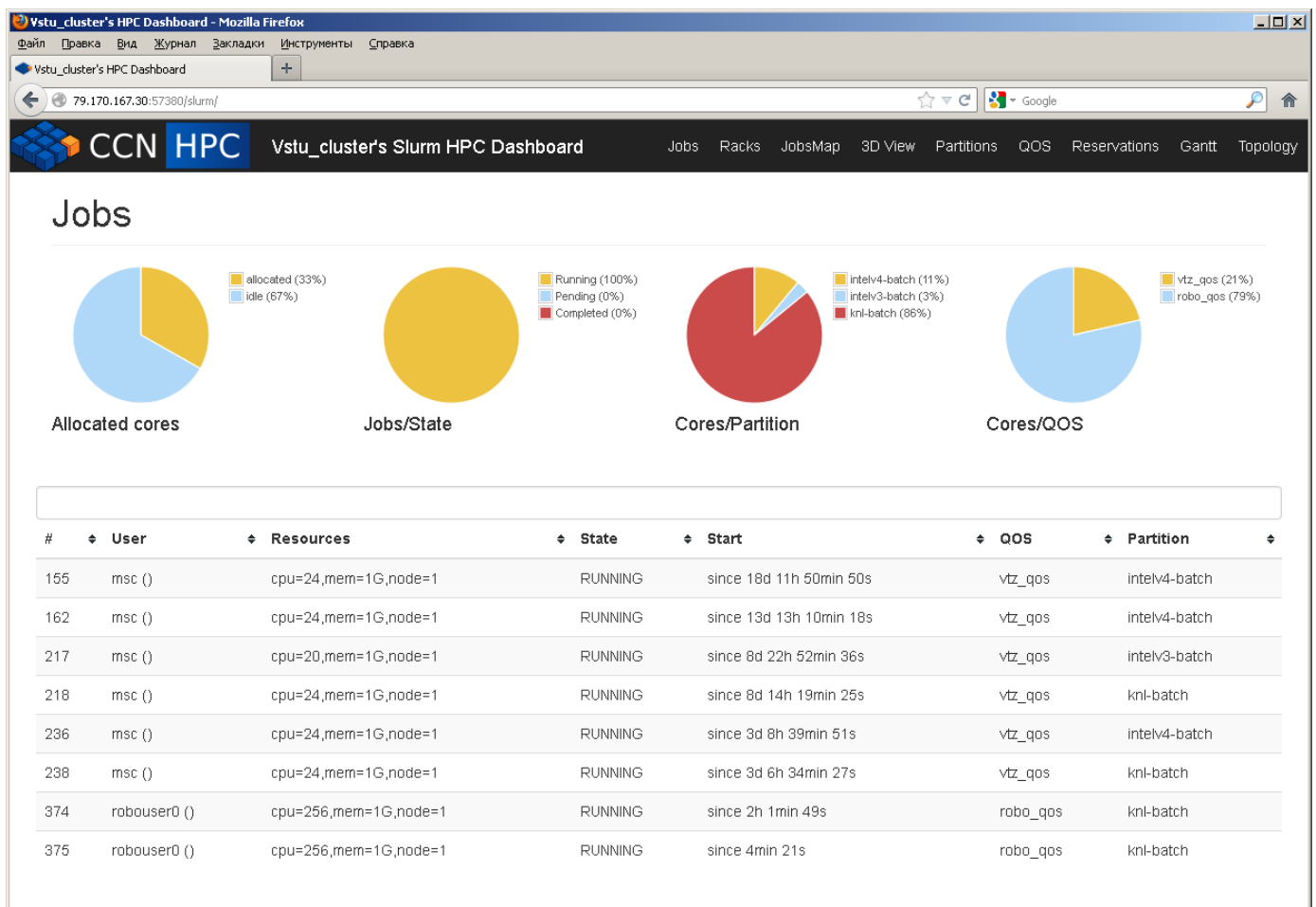
[robouser0@headnode opt]$
[robouser0@headnode lstm_model]$ sbatch start.sh
Submitted batch job 375
[robouser0@headnode lstm_model]$ squeue
JOBID PARTITION NAME USER ST TIME NODES MODELIST(REASON)
155 intelv4-b post_ben msc R 18-11:46:57 1 node52.cluster
162 intelv4-b post_ben msc R 13-13:06:25 1 node51.cluster
217 intelv3-b form1153 msc R 8-22:48:43 1 node40.cluster
218 knl-batch post_ben msc R 8-14:15:32 1 node32.cluster
236 intelv4-b post_ben msc R 3-08:35:58 1 node53.cluster
238 knl-batch post_ben msc R 3-06:30:34 1 node32.cluster
374 knl-batch ivanov_1 robouser R 1:57:56 1 node34.cluster
375 knl-batch lstm exa robouser R 0:28 1 node35.cluster
[robouser0@headnode lstm_model]$ █
```

10. После запуска программы в системе очередей на исполнение в директории `lstm_model` появится лог-файл, в который будет записываться вывод программы. При необходимости, его содержимое можно просмотреть, нажав клавишу **F3** (выйти из режима просмотра текста можно с помощью клавиши **esc**).



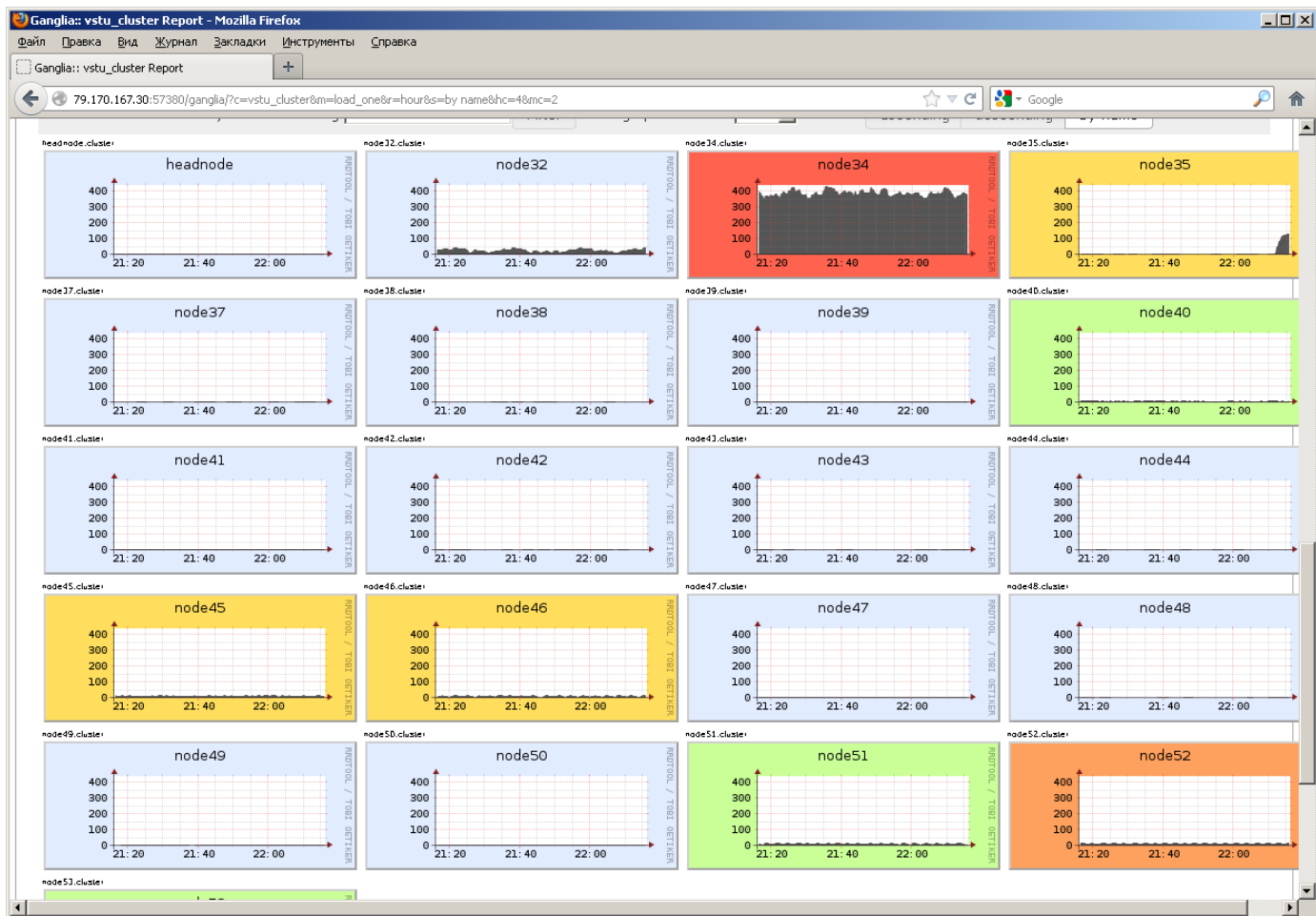
11. Статус задачи в системе очередей можно узнать также с использованием web-интерфейса, перейдя в браузере по адресу:

<http://79.170.167.30:57380/slurm>



12. Текущую загрузку вычислительных узлов можно посмотреть с помощью системы мониторинга Ganglia, перейдя в браузере по адресу:

<http://79.170.167.30:57380/ganglia>



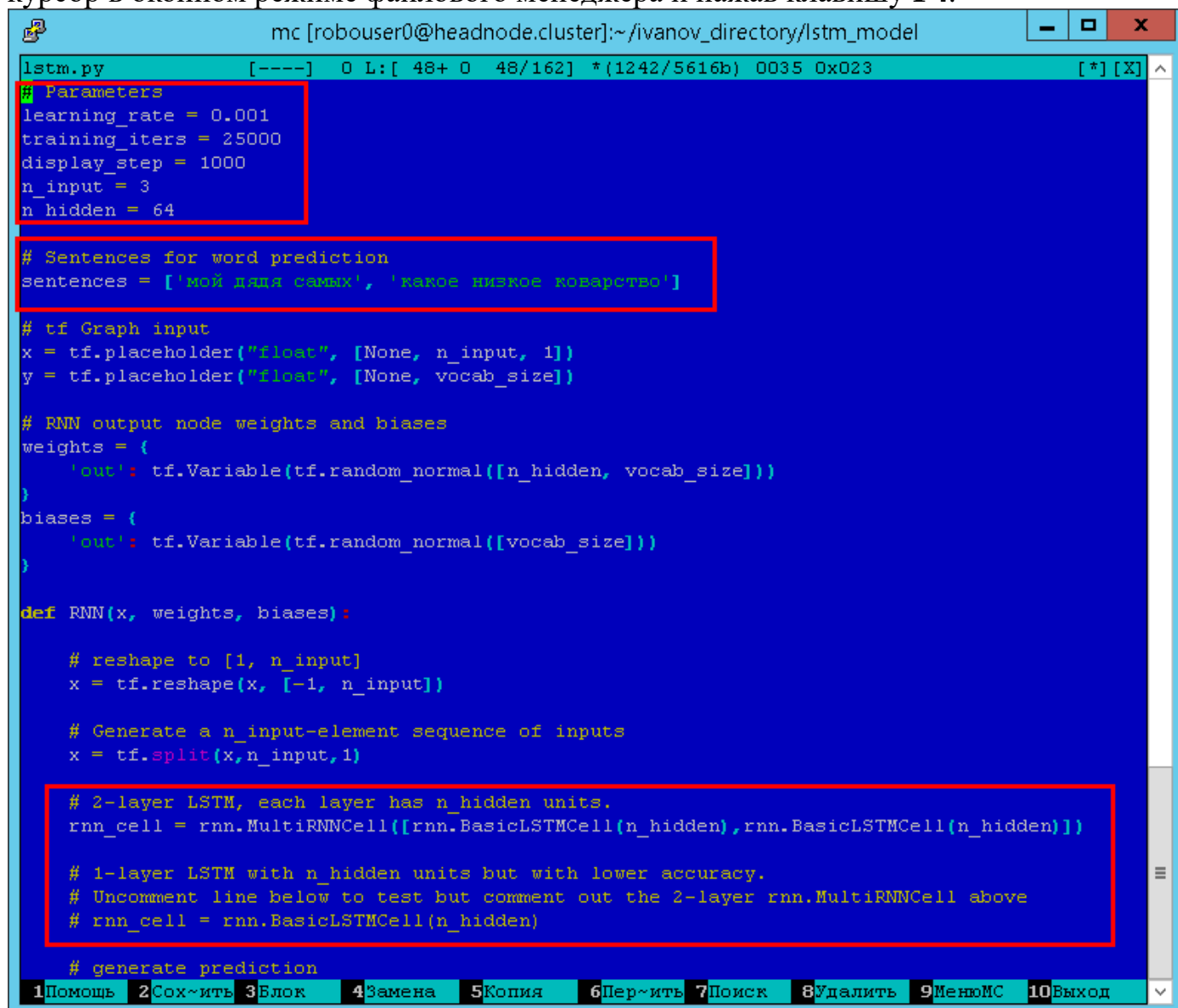
13. После того, как обучение модели будет завершено, начнется процесс предсказания последовательности слов. В качестве исходной последовательности берутся строки из трех слов, заданные в переменной `sentences` в файле `lstm.py`. После завершения работы программы (это можно узнать по статусу соответствующей ей задачи в `slurm`) результат предсказания можно посмотреть в лог-файле. Содержимое лог-файла позволяет определить расчетное время и точность обучения или содержимое ошибок в случае неуспешного завершения задачи:

```

mc [robouser0@headnode.cluster]:~/ivanov_directory/lstm_model
/home/robouser0/ivanov_directory/lstm_model/lstm-log-384.txt 6616/6616 100% ^
['волею', 'зевеса', 'наследник'] - [всех] vs [всех]
Iter= 20000, Average Loss= 0.122205, Average Accuracy= 95.90%
['читатель', 'там', 'некогда'] - [гулял] vs [гулял]
Iter= 21000, Average Loss= 0.080570, Average Accuracy= 97.20%
['правил', 'когда', 'не'] - [в] vs [в]
Iter= 22000, Average Loss= 0.063658, Average Accuracy= 97.90%
['низкое', 'коварство', 'полуживого'] - [забавлять] vs [забавлять]
Iter= 23000, Average Loss= 0.065370, Average Accuracy= 97.60%
['своих', 'родных', 'друзья'] - [людились] vs [людились]
Iter= 24000, Average Loss= 0.068918, Average Accuracy= 97.70%
['гулял', 'и', 'я'] - [но] vs [но]
Iter= 25000, Average Loss= 0.069387, Average Accuracy= 97.60%
['какая', 'скука', 'с'] - [большим] vs [большим]
Optimization Finished!
Elapsed time: 5.862200593948364 min
Model successfully restored!
Predictions:
мой дядя самых честных правил когда не в
какое низкое коварство своих родных друзья людились и
1Помощь 2Разверн 3Выход 4Нех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход

```

17. Результат предсказания модели зависит от длины текста, параметров модели, числа слоев нейросети и т.д. Видно, что результат предсказания неточен, поскольку для ускорения обучения нейросети была использована слишком грубая модель. Для уточнения параметров модели можно поправить файл `lstm.py`, наведя на него курсор в оконном режиме файлового менеджера и нажав клавишу **F4**.



```
mc [robouser0@headnode.cluster]:~/ivanov_directory/lstm_model
lstm.py [----] 0 L:[ 48+ 0 48/162] *(1242/5616b) 0035 0x023 [*] [X] ^
# Parameters
learning_rate = 0.001
training_iters = 25000
display_step = 1000
n_input = 3
n_hidden = 64

# Sentences for word prediction
sentences = ['мой дядя самый', 'какое низкое коварство']

# tf Graph input
x = tf.placeholder("float", [None, n_input, 1])
y = tf.placeholder("float", [None, vocab_size])

# RNN output node weights and biases
weights = {
    'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
}
biases = {
    'out': tf.Variable(tf.random_normal([vocab_size]))
}

def RNN(x, weights, biases):

    # reshape to [1, n_input]
    x = tf.reshape(x, [-1, n_input])

    # Generate a n_input-element sequence of inputs
    x = tf.split(x, n_input, 1)

    # 2-layer LSTM, each layer has n_hidden units.
    rnn_cell = rnn.MultiRNNCell([rnn.BasicLSTMCell(n_hidden), rnn.BasicLSTMCell(n_hidden)])

    # 1-layer LSTM with n_hidden units but with lower accuracy.
    # Uncomment line below to test but comment out the 2-layer rnn.MultiRNNCell above
    # rnn_cell = rnn.BasicLSTMCell(n_hidden)

    # generate prediction
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС 10Выход
```

19. После правки параметров удалите файлы сохраненной модели и запустите обучение вновь. Имейте в виду, что чем точнее модель, тем дольше она будет обучаться. Можно взять другой текст для обучения. После получения обученной модели проведите предсказание слов и сравните качество предсказания с предыдущим примером. Сделайте выводы.