Moa

0.1

Author: Mark Fiers

Date: Tue Jul 21 10:22:56 NZST 2009

Git: master - 57ffd07addfb58357071c1f6094e13280ca8b65a

Contents

Chapter 1

Introduction

Kea is a simple, but powerful, script automation system. It is useful for automating small scripts that need to be executed regularly, or need to be repeated many times, for example for a set of different input files.

For example, a BLAST analysis [1], you could perform a single analysis using the following command:

```
blastall -i seq1.fasta -o blast1.out -p blastn -d nr
```

Now suppose you want to repeat this for a set of sequences. This is, again, not very difficult using the following bash script:

```
for in in /data/seq/*.fasta; do
  blastall -i $in -p blastn -d nr \
      -o `basename $in .fasta`.out
done
```

But what if you would like to repeat this weekly, for several data sets, against a few databases, but only for those sequences that have changed? Furthermore, the sequences come from different sources with different naming strategies and you would like uniformly named output? It can all be done using bash, or any other scripting language, but, you will write a lot of boilerplate code. Kea can help you.

1.1 Kea

To get started, we'll set up a job with Kea executing the sample from above. But first, lets introduce Kea basics.

Kea automates the execution of (small) scripts. Each script automated by Kea is called a Kea job. The author proposes that you store in and output files in separate directories with the Kea jobs described in the directories where the output files are stored. Kea itself does not make such a demand. Kea assumes that for each job the input and output files are stored in separate directories. A Kea Job is typically linked to the directory where the results of the job are

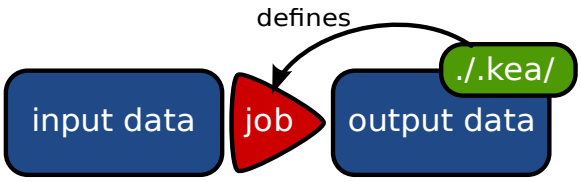

Figure 1.1: Basic structure of a Kea job; A job is usually defined in the directory where the results end up.

stored (see 1.1. A Kea job is linked to a directory by storing the configuration in a sub-directory called `./.kea`

Linking a Kea job to a single directory, allowing only one job per directory is a conscious design decision. It keeps jobs organized and makes execution and re-execution of jobs simple (as you will see later), without putting actual limits on what you can do.

Back to our earlier example. Suppose that the original sequences live in a directory called `/data/seq/` and the blast results are to end up in `/data/blast`. The first step would be to create and move to the target directory[1]:

```
mkdir /data/seq
cd /data/seq
```

Secondly, create a basic kea configuration using the following command:

```
kea new
```

This command creates a `.kea` sub-directory (if it doesn't exists) and copies a basic configuration file. This configuration file is named `./.kea/local.conf` and controls what exactly your job will do. Once you open it in an text editor it looks like a normal configuration file. It is divided up in sections that each have a number of options. Currently you can ignore most of these options. To get our blast running, there are two sections that need some work. First we'll define the BLAST input in, suprisingly, the `[INPUT]` section. If you using the directories suggested earlier, set the `[INPUT]` section to:

```
[INPUT]
glob=/data/seq/*.fasta
```

or, depending on where you're target directory is, and thus where you have defined your current Kea job, you could define the input section using relative locations:

```
[INPUT]
glob=../seq/*.fasta
```

[1] I find it difficult to estimate the right level of detail. Although I do not expect that anybody able to operate Kea really needs to be told how to use cd or `mkdir`, but I prefer to add to much details rather than too little

Using relative directory locations makes it easy to shuffle your jobs around, as long as their location relative to each other doesn't change.

The second part of the configuration that needs editting is the `[ACTION]` section. This section defines what needs to be done. For this example, change it to:

```
[ACTION]
main = blastall -i $inputFile -d nr -p blastn -o $baseName.out
```

This defines one item (main) in the action section. It is possible to have scripts of multiple lines, for example, if you would like to print some extra information, you could extend your configuration like this:

```
[ACTION]
main =
  echo "start␣with␣$inputFile"
  time blastall -i $inputFile -d nr -p blastn -o $baseName.out
  echo "$inputFile␣has␣finished"
```

To make sure the configuration file is parsed properly, it is important that there is a space in front of each line of the script. The main item is extended until there is a character in the first column (where it is assumed that a new item starts). This allows longer scripts.

The attentive reader might have noticed that there are two, as of yet, unexplained variables in the scripts. These are provided by Kea.

In general terms, you give Kea a script and define some conditions. Kea subsequently executes the script as many times as necessary. All communication between Kea and the script go through environment variables.

In the example above, Kea determines how many `*.fasta` files there are in `/data/seq` and iteratively executes the script once for each of those inputfiles with the full path of each ddinput file stored in the `$inputFile` environment variable.

asdf

Chapter 2

Installation

.. to be written ..

2.1 Getting the code

2.2 Install python libs

2.3 Set up a basic configuration

2.4 Test

Chapter 3

Configuration

3.1 the .kea directory

3.2 Basic configuration

3.2.1 JOB

3.2.2 PLANNER

3.2.3 ACTOR

3.2.4 INPUT

3.2.5 ACTION

3.3 Templates

3.4 Learn to fly

Bibliography

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, Oct 1990.