Программный конвейер Маши

Маша, биолог, только что вернулась из шестимесячной командировки (Баренцево море), где она собирала образцы. Всего у нее ~1500 образцов, и теперь ей необходимо:  
 1. Пропустить каждую пробу через аналитическую машину, которая будет измерять относительное содержание 300 различных белков. Результатом работы машины для одного образца является файл, где каждому белку будет сопоставлена одна строка.  
 2. Рассчитать статистику для каждого из белков, используя программу, написанную ее руководителем под названием protstats.  
 3. Описать полученные результаты. Ее руководитель хотел бы, чтобы она сделала это к концу месяца, так чтобы ее статья могла появиться в предстоящем специальном выпуске специализированного журнала.  
Маше потребуется около двух недель, чтобы пропустить образцы через аналитическую машину. После чего, перед ней будет стоять грандиозная задача: проанализировать полученные результаты, запустив protstats. Плохая новость заключается в том, что если вручную запускать protstats с помощью графического интерфейса, то Маше придется воспользоваться диалоговым окном открытия файла ~1500 раз и в сумме весь процесс займет более 12 часов (предполагая, что она будет готова выбрать следующий файл, как только завершится предыдущий анализ, и учитывая что один образец будет считаться 30 секунд). На самом деле такой сценарий может быть реализован, только если Маша — робот, поэтому, вероятно, анализ займет гораздо больше 12 часов, не говоря уже о том, что шансы на правильный выбор всех этих файлов практически равны нулю. Таким образом, пропуск сроков подачи статьи выглядит все более вероятным.

Следующие несколько страниц будут посвящены тому, как на самом деле Маше следуют поступать. Если более конкретно, будет показано, как Маша может использовать bash для запуска программы protstats, используя циклы для автоматизации повторяющихся шагов, например, ввод имен файлов, так чтобы ее компьютер мог работать 24 часа в сутки, пока она пишет статью.

В качестве бонуса, после того, как она соберет скрипты обработки в конвейер, у Маши появится возможность использовать конвейер снова, когда соберет больше данных.

Для начала Маше необходимо организовать правильным образом данные. Для этого, она, первым образом, создает каталог под названием barents-sea-samples (чтобы напомнить себе, откуда поступили данные). Внутри этого каталога она создает другой - с названием 2019-05-13, название которого есть отсылка к дате начала обработки образцов. Раньше она использовала другие названия, такие как conference-paper и results, но, через пару лет, ей становилось трудно понять к чему они относились (по прошествии какого-то времени сложно вспомнить к чему относится, например, каталог revised-revised-results-3).

**Сортивка вывода результатов**

Маша использует в названиях директорий последовательность год-месяц-день с добавлением нулей, по причине того, что bash выводит названия файлов и директорий в алфавитном порядке. Если бы она не придерживалась такого формата, то, например December шел бы перед , например July (если бы Маша писала названия месяцев), или 11 (Ноябрь) шел бы перед 2 (Февраль).

Каждый образец маркирован в соответствии с соглашением принятым в ее лаборатории -

«MA01729А», поэтому Маша решает использовать такой ID в названии файлов MA01729A.txt, MA01812A.txt и так далее. Все 1500 файлов будут находиться в одной директории.

Итак, находясь в Машиной исходной директории data-shell, можно посмотреть список файлов командой:

$ ls barents-sea-samples/2019-05-13/

Довольно много писать, но время написания можно сократить если использовать функцию автозаполнения:

$ ls bare

и затем нажать клавишу Tab оболочка автоматически допишет полное имя директории:

$ ls barents-sea-samples/

Если нажать на клавишу Tab снова – bash добавит директорию 2019-05-13 в текст команды, так как других возможностей для автозаполнения нет, в противном случае автозаполение не сработает. Однако, двойное нажатие на клавишу Tab выведет все возможные варианты автозаполнения.

**Проверка файлов**

Маша “прогнала” образцы через аналитические машины и получила на выходе 17 файлов, находящихся в директории barents-sea-samples/2019-05-13. В качестве быстрого теста Маша запускает вот такую команду:

$ cd barents-sea-samples/2019-05-13

$ wc -l \*.txt

300 MA01729A.txt

300 MA01729B.txt

300 MA01736A.txt

300 MA01751A.txt

300 MA01751B.txt

300 MA01812A.txt

… …

Теперь, Маша набирает вот такую команду:

$ wc -l \*.txt | sort -n | head -n 5

240 MA02018B.txt

300 MA01729A.txt

300 MA01729B.txt

300 MA01736A.txt

300 MA01751A.txt

В одном из файлов недостает 60 строк! Когда Маша поднимает историю этого файла — оказывается, что она запускала этот образец в понедельник утром и, возможно, аналитическая машина работала все выходные, а Маша забыла ее перезагрузить. Перед тем как отправить образец на повторный анализ, она снова проверяет файлы — нет ли в них таких, в которых было бы больше данных чем положено.

$ wc -l \*.txt | sort -n | tail -n 5

240 MA02040B.txt

300 MA02040Z.txt

300 MA02043A.txt

300 MA02043B.txt

5040 total

Теперь числа выглядят хорошо — но в окончании одного из файлов появилась литера Z. По конвенции принятой в ее лаборатории, такая литера обозначает недостоверный результат. Маше нужно проверить — нет ли еще таких файлов, потому что в последующем анализе они участвовать не должны.

$ ls \*Z.txt

MA01971Z.txt MA02040Z.txt

Маша могла бы просто стереть эти файлы командой rm, но, с другой стороны в анализе, где недостоверность данных не имела бы значения они могли бы быть полезны, поэтому Маша решает их оставить и остальные операции проводить над подвыборкой файлов используя выражение \*[AB].txt. В таком выражение \* означает любые символы, а [AB] — только литеры А и В, делая подвыборку корректной.

**Обработка файлов**

Наконец-то Маша готова к запуску программы protstats — shell скрипт написанный ее начальником. Скрипт вычисляет некоторые статистические параметры и принимает на вход два агрумента:

1. входной файл
2. выходной файл(куда будут записаны результаты)

Так Маша все ее обучается премудростям написать shell скриптов она решает составить нужную команду последовательно. На первом этапе она должна убедиться, что выделяет необходимые файлы правильно (файлы с литеров А или В, игнорируя файлы с литерой Z). Начиная из корневой для проекта директории, Маша пишет следующие команды:

$ cd barents-sea-samples/2019-05-13

$ for datafile in MA\*[AB].txt

> do

> echo $datafile

> done

MA01729A.txt

MA01729B.txt

MA01736A.txt

...

MA02043A.txt

MA02043B.txt

Следующим шагом Маша решает как назвать выходные файлы, полученные в результате работы программы protstats. Она решает добавить префикс “stats-”.

$ for datafile in MA\*[AB].txt

> do

> echo $datafile stats-$datafile

> done

MA01729A.txt stats-MA01729A.txt

MA01729B.txt stats-MA01729B.txt

MA01736A.txt stats-MA01736A.txt

...

MA02043A.txt stats-MA02043A.txt

MA02043B.txt stats-MA02043B.txt

Надо заметить, что Маша пока что не запускала программу, но теперь она может быть уверена, что будут выделены нужные файлы и опять же нужные файлы будут созданы.

Набор команд раз за разом может быть утомительным и приводить к ошибкам. Поэтому, чтобы запустить обсчет, Маша, вместо того, чтобы заново писать цикл нажимает на клавишу вверх. В результате в командной строке появляется следующая команда (результат агрегирования цикла):

$ for datafile in MA\*[AB].txt; do echo $datafile stats-$datafile; done

Используя курсор Маша меняет echo на bash protstats:

$ for datafile in MA\*[AB].txt; do bash protstats $datafile stats-$datafile; done

Теперь, когда она нажимает Enter, bash начинает обрабатывать модифицированную команду. Однако на экране ничего не происходит — непонятно идет ли программа, как быстро… Маша прерывает обработку нажав комбинацию клавиш Ctrl-C, и модифицирует команду :

$ for datafile in MA\*[AB].txt; do echo $datafile; bash protstats $datafile stats-$datafile; done

**История**

Еще один способ повторения набранного когда-либо выражения использовать команду history чтобы вывести список уже выполненных команд и затем использовать !123 (где “123” должно быть заменено нужным числом). Например, Маша могла бы написать:

$ history | tail -n 5

456 ls -l MA0\*.txt

457 rm stats-MA01729B.txt.txt

458 bash protstats MA01729B.txt stats-MA01729B.txt

459 ls -l MA0\*.txt

460 history

Теперь можно повторить анализ MA01729B.txt просто набрав !458.

Существуют и другие быстрые команды для работы с history:

* Ctrl-R позволяет пользоваться историей в стиле контекстного поиска предоставля последнюю использованную команду с таким словосочетанием. Дополнительно нажатие Ctrl-R позволяет посмотреть более ранние команды.
* !$ позволяет получить последнее слово последней исполненной команды.

**Написание собственных скриптов**

Руководитель Маши, настаивает, чтобы все что она делала было воспроизводимым. Один из легких путей для этого — написание собственных скриптов.

Для начала надо переместиться в директорию с данными.

$ cd ../barents-sea-samples/2019-05-13

Затем, Маша запускает редактор и пишет следующее:

# Calculate stats for data files.

for datafile in "$@"

do

echo $datafile

bash protstats $datafile stats-$datafile

done

Она сохраняет скрипт в в файле do-stats.sh, так что теперь она может воспроизвести первую часть анализа написать в терминале:

$ bash do-stats.sh MA\*[AB].txt

Кроме того, она может написать вот так:

$ bash do-stats.sh MA\*[AB].txt | wc -l

так что на выход будет выведены число обработанных файлов заместо их имен.