*М.Р. Лапшин, асп.; рук. С.Т. Лескин, д.т.н, проф.;*

*рук. А.О. Скоморохов д.т.н, проф.*

**(ИАТЭ НИЯУ МИФИ, г. Обнинск)**

Автоматизированная подготовка данных для проведения диагностики состояния оборудования АЭС

Для обеспечения безопасной и экономичной работы АЭС применяются системы контроля оборудования, которые состоят из нескольких тысяч датчиков. Объем поступающей информации может достигать сотни гигабайт в год. С помощью специально поставленного технического и программного обеспечения полученная информация архивируется на соответствующих серверах, которые находятся непосредственно на АЭС. Данные накапливаются и не анализируются на предмет выявления совокупных зависимостей элементов оборудования друг от друга, выявления неочевидных и скрытых тенденций развития аномалий.

Для анализа состояния технического оборудования соответствующие данные извлекаются из архивов. Извлеченные файлы могут быть как в формате специализированных библиотек, так и в простом текстовом формате. Передача данных в простом текстовом формате исключает необходимость иметь у исследовательских организаций специализированное программное обеспечение для обработки файлов. Но остается вопрос об иснтрументах работы с текстовыми файлами размером 5 Гб и более.

Данная работа посвящена обзору инструментов на базе систем *LINUX/UNIX*, которые позволяют свободно обрабатывать текстовые файлы больших размеров. Рассмотривается первый этап анализа данных - подготовка и приведение снятых с приборов показаний в удобный для последующего детального анализа вид.

Имеются данные технологического контроля с четырех ГЦН (главный циркуляционный насос) 3-го блока Калининской АЭС, записанные в файл *data.txt*. За состоянием всех составляющих насоса ведут постоянный контроль, так как останов ГЦН вследствии выхода из строя приведет к большим экономическим потерям. На каждый ГЦН устанавливается свыше 50 датчиков контроля с различных систем.

Для проведения дальнейшего детального анализа необходимо, чтобы данные были приведены к определенному виду, т.е. все сигналы должны содержаться в отдельных файлах и иметь определенную структуру каталогов. Нижеприведенная табл.1 дает общее представление о содержании файла *data.txt*.

Таблица 1 – Cодержание файла data.txt

| **Название** | **дата-время** | **значение** | **...** | **служеб.параметры** |
| --- | --- | --- | --- | --- |
| гцн1\_датчик1 | 2011-01-01 00:00:00 | 285,023 | ... | ... |
| гцн1\_датчик2 | 2011-01-01 00:00:01 | 34,8562 | ... | ... |
| гцн3\_датчик3 | 2011-01-01 00:01:01 | NULL | ... | ... |
| общ\_параметр1 | 2011-01-01 00:01:01 | 2998,84 | ... | ... |
| гцн2\_датчик2 | 2011-01-01 00:01:02 | 34,4822 | ... | ... |
| ... | ... | ... | ... | ... |
| гцн4\_датчик33 | 2013-04-03 13:01:56 | 54,0454 | ... | ... |

Сложность проведения анализа данных заключается в следующем:

* представление данных одним текстовым файлом большого размера (6.2 Гб ~ 38.5 млн строк);
* запись сигналов со всех ГЦН и общеблочных параметров друг за другом в одном файле (как показано в таблице 1);
* различная частота опроса датчиков;
* рассинхронизация сигналов относительно друг друга;
* непоследовательная по времени запись сигналов;
* неудобный формат времени;
* лишние для проведения анализа столбцы;
* десятичный разделитель в виде запятой;
* наличие значений *NULL* (поля в базе данных, не имеющее никакого значения, но записанные в файл);
* большое количество выбросов.

Далее приводится набор инструментов *LINUX*, которые использовались для обработки данных.

## 1. Cat, head, tail

*Cat* - утилита, выводящая в терминал указанный файл(ы); *head*/*tail* - утилиты, выводящие первые/последние n строк из файла. Используя данные утилиты, можно получить общее представление о содержании файла. В работе утилита *cat* использовалась с перенаправлением вывода в программу просмотра файлов *less*. Примеры применения утилит приведены ниже.

***cat data.txt | less***

**2. Sed** - это потоковый текстовый редактор, который позволяет изменять каждую строку файла по определенному правилу. Пример: замена десятичного разделителя с запятой на точку без перенаправления в другой файл.

***sed -i 's/,/./g' data.txt*** (*s* - замена, *g* - замена во всем файле).

## 3. Cut - команда выборки отдельных полей из строк файла. В случае файла data.txt столбцы разделены с помощью tab, и поэтому 1-3 столбцы, необходимые для дальнейшего анализа, можно вырезать без указания разделителя:

***cut -f 1,2,3 data.txt > 1\_3.txt***

**4. Grep** - утилита командной строки, которая действует по принципу "искать и выводить строки, соответствующие регулярному выражению". Пример: поиск и вырезание сигналов *null*.

***grep 'NULL' 1\_3.txt > null.txtgrep -v 'NULL' 1\_3.txt > 1\_3\_nonull.txt***

Первое выражение нужно для того, чтобы удостовериться, что значения *NULL* существуют в файле. А второе - оставляет строки, которые не содержат значения *NULL*. Таким же образом можно избавиться от всех ненужных строк. Полученный файл *1\_3\_nonull.txt* представляет собой таблицу с тремя столбцами (название датчика, дата-время, значение).

## 5. Использование скриптов

Для дальнейшего разбиения файла на отдельные части, в которых будут содержаться сигналы с определенного датчика, используется скрипт, написанный на языке *bash*.

***mkdir -p $3***

***for name in $(cat $1); do***

***var1=$(echo $name | cut -d';' -f 1)***

***var2=$(echo $name | cut -d';' -f 3)***

***grep -w $var1 $2 > $3/${var2}.txtdone***

Пример использования:

***. multigrep.sh list.txt 1\_3\_nonull.txt gcn/g1/temp/1\_3*,**

где *". multigrep.sh"* - вызов скрипта; первый аргумент *list.txt* - файл со списком датчиков, которые мы хотим отделить и положить в определенный каталог (например температуры подшипников по первому ГЦН); второй - 1\_3\_nonull.txt - данные; третий - каталог, в который будут сложены файлы.

Теперь в каталоге *gcn/g1/temp/1\_3* лежат все датчики температур по 1 - ГЦН отдельно по файлам. Каждый файл имеет удобное название и представляет собой таблицу из трех столбцов: аббревиатура датчика (одинаковая во всех строках), дата-время, значение. Далее необходимо вырезать 2 и 3 столбцы из полученных файлов по каждому датчику, для чего используется скрипт *multicut*:

***. multicut.sh gcn/g1/temp/1\_3 2 gcn/g1/temp/2***

***. multicut.sh gcn/g1/temp/1\_3 3 gcn/g1/temp/3***

Таким образом, получаем набор файлов, в которых содержатся только 2 и 3 столбцы.

## 6. Перевод из формата "datatime" в формат "секунды"

Для перевода из формата "datatime" в формат "секунды" используется скрипт *multidate*. Первый аргумент - каталог с файлами, которые нужно преобразовать; второй - точка отсчета; третий - каталог назначения.

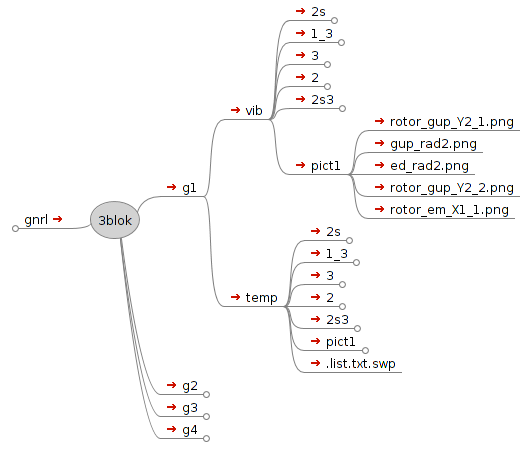
***. multidate.sh gcn/g1/temp/2 "2011-03-05 00:00:00" gcn/g1/temp/2s***

## 7. Создание таблицы из нескольких "одностолбцовых" файлов

Чтобы получить полноценный временной ряд, необходимо объединить файлы со значениями времени измерения в секундах с файлами, в которых содержатся значения сигналов. Для этого используется скрипт *paste*, который рассматривает файлы, как вертикальные колонки, соединяет их и выводит в файл.

***. multipaste.sh gcn/g1/temp/2s gcn/g1/temp/3 gcn/g1/temp/2s3***

## 8. Построение множества графических зависимостей

Для построения графиков использовался скрипт с программой *gnuplot* - свободная программа для создания двух- и трехмерных графиков.

***.multiplot.sh gcn/g1/temp/2s3 gcn/g1/temp/pict1 my.gp***

Теперь в директории *gcn/g1/temp/pict1* располагаются графики. Общая структура полученных каталогов представлена на рис. 1.

Рис. 1 - Общая структура каталогов

Как было показано выше, сложные операции с файлами совершаются в "один удар" с помощью определенных команд, утилит и небольших скриптов. Это облегчает процесс подготовки данных и позволяет подвести их к следующему этапу анализа в нужном виде. Дальнейшие действия, такие как синхронизация, устранение выбросов и проведение анализа реализовываются на языке программирования APL.

# Библиографический список

1. Зарюгин Д.Г. Разработка алгоритмов диагностики состояния ГЦН АЭС с ВВЭР-1000 по данным оперативного технологического контроля. Обнинск. 2001.

2. Mendel C. Advanced Bash-Scripting Guide. 2012.

3. Граннеман С. Linux. Необходимый код и команды. Москва. 2010.

4. Philipp K.Janert. Gnuplot in action. 2010.