Инструкция по созданию дампов Firebird на Linux

Любое программное обеспечение содержит ошибки. Также и СУБД Firebird – не застрахована от ошибок. Конечно, Firebird регулярно и тщательно тестируется, но абсолютно все ситуации предусмотреть невозможно. Кроме того, есть ошибки разного рода. Например, некоторые ошибки можно воспроизвести последовательностью команд SQL или программного кода приложения. А некоторые ошибки проявляются весьма редко, и могут быть связаны со структурой таблиц, конкретной операционной системой, и другими специфическими условиями. Единственный способ поиска таких ошибок – анализ дампов памяти, произведенных на конкретном компьютере для конкретного экземпляра Firebird.

Чтобы получить дамп, и передать его разработчикам для исследования, необходимо выполнить ряд действий, о которых пойдет речь ниже.
(аналогичная статья по дампам на Windows – здесь)

! iBase.ru осуществляет анализ дампов только как часть действующей технической поддержки для предприятий.

Внимание! Все нижеперечисленные действия следует выполнять только с правами root'а.

Остановка/удаление пакетов, препятствующих созданию дампов

1) **Ubuntu**

1.1) проверить и удалить пакет systemd-coredump.
Ввести
```
cat /proc/sys/kernel/core_pattern
```

Если в ответ будет выдано что-то типа `|/lib/systemd/systemd-coredump %P %u %g %s %t %e` - удалить его:
```
apt-get purge --auto-remove systemd-coredump
```

(см также тут: https://www.howtoinstall.co/en/ubuntu/xenial/systemd-coredump?action=remove)

1.2) проверить и удалить пакет apport

- поиск пакета: `service --status-all | grep -m1 -i apport`
- проверка допустимости (симуляция) удаления: `apt-get -s remove apport`
- действительное удаление: `sudo apt-get remove apport`
- очистка всех "хвостов": `sudo apt-get purge apport`

2) **CentOS/RHEL**

2.1) Проверить и удалить пакет **abrtd**

Ввести `cat /proc/sys/kernel/core_pattern`

Если будет выдано что-то типа `|/usr/libexec/abrt-hook-ccpp %s %c %p %u %g %t e %P %I` - остановить и удалить этот пакет:

- остановка: a) **centos-6**: `service abrtd stop` ; b) **centos-7**: `systemctl disable abrtd.service`
- удаление: `yum -y remove abrt*`
- очистка "хвостов": `rm -r /var/cache/abrt/`
- проверка того, что всё было удалено: `cat /var/log/yum.log | grep abrt`

Снятие ограничений на размер создаваемых дампов

Прежде всего, надо убедиться, что система вообще допускает создание дампов процессов при их падении.
Вводим команду: `ulimit -c`
Если там в ответ будет "0" (что скорее всего для RH/Centos), то исправляем это ограничение:

1) в системах до CentOS-6 включительно использовался старый init-загрузчик ("SysV").
Для снятия лимитов в таких ОС нужно:

- открыть на редактирование файл `/etc/security/limits.conf`
- добавить туда строку: `* soft core unlimited`

2) в системах, начиная с CentOS-7, а также в Ubuntu, используется init-загрузчик "systemd".
Здесь наличие строки `* soft core unlimited` (в файле `/etc/security/limits.conf`) нужно
только для создания дампов при падениях клиентских утилит (isql, fbsvcmgr, gbak, ...), а также когда firebird запущен как приложение,а не сервис.

Для обычного варианта работы Firebird (как «сервис») эта строка в "systemd"-загрузчике ни на что не влияет, надо менять скрипт запуска службы:

- CENTOS: открыть в редакторе файл `/usr/lib/systemd/system/firebird-superserver.service`
- UBUNTU: открыть в редакторе файл `/lib/systemd/system/firebird-superserver.service`
- и добавить там в секцию `[Service]` параметр:

```
LimitCORE=infinity
```

Затем запустить команду
```
systemctl daemon-reload ; systemctl restart firebird-superserver.service
```

После рестарта службы следует убедиться, что процессу Firebird предоставлен неограниченный лимит на размер core-файлов.
Для этого можно использовать команду:

```
strings /proc/$(pgrep firebird)/limits
```

Пример её вывода:

```
Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
...
Max core file size        unlimited            unlimited            bytes
...
```

Задание шаблона для имён создаваемых дампов

Далее надо указать параметры имени создаваемого core-файла.
По умолчанию файл будет иметь вид "core" или "core.NNN" (где NNN = pid упавшего процесса), что неудобно с точки зрения администратора: в имени дампа не содержится полный путь и имя бинарника, который упал.

Чтобы добавить в шаблон coredump'a название бинарника упавшего процесса, следует добавить **%e** или **%E** (в первом случае будет только имя файла без пути, во втором - путь, в котором прямой слеш заменён на знак восклицания).
Чтобы добавить число секунд, прошедших с 01.01.1970 00:00:00, следует добавить **%t** (к сожалению, нельзя заставить выводить момент краша в формате типа yyyymmdd_hhmmss.zzz).
Чтобы добавить PID упавшего процесса, следует использовать **%p**.
Для практики лучше сразу после признака **%e** прописывать признак штампа времени **%t**, и только после него - номер PID'a процесса (**%p**).
Так обеспечится сортировка имен дампов по времени при просмотре каталога встроенной командой `ls -l`.

Дополнительная информация.

- Значения букв после знака процента в шаблоне имени файла дампа (http://man7.org/linux/man-pages/man5/core.5.html):
- `%c  core file size soft resource limit of crashing process (since Linux 2.6.24)`
- `%h: hostname`
- `%e: executable filename (without path prefix)`
- `%t: UNIX time of dump`
- `%p: pid`
- `%E  pathname of executable, with slashes ('/') replaced by exclamation marks ('!') (since Linux 3.0)`
- `%s: signal number`

- Максимальная длина имени core-файла составляет 128 байт.

https://linux-audit.com/understand-and-configure-core-dumps-work-on-linux/

Способ, которым следует менять шаблон имени core-файлов, зависит от init-загрузчика текущей OS (старый: "SysV" (или его подвид: "upstart") или современный: "systemd").
1) в загрузчике *systemd* следует:

- задать значение в рамках текущего сеанса:

```
sysctl -w kernel.core_pattern=/каталог/core.%e.%t.%p
```

- для фиксации этого значения (чтобы оно оставалось постоянным после рестарта сервера) - добавить в файл `/etc/sysctl.conf` строку:

```
kernel.core_pattern=/каталог/core.%e.%t.%p
```

2) в старом загрузчике SysV (использовался до CentOS 6.9 ключительно):

- задать значение в рамках текущего сеанса: `echo "/каталог/core.%e.%t.%p" > /proc/sys/kernel/core_pattern`
- для фиксации этого значения (чтобы оно оставалось постоянным после рестарта сервера) - добавить в файл `/etc/rc.d/rc.local` эту же строку
    - и указать для этого файла признак его исполняемости: `chmod +x /etc/rc.d/rc.local`

В любом случае, после этого - НЕ ЗАБЫТЬ ПОМЕНЯТЬ ВЛАДЕЛЬЦА У "/каталог":
`chown firebird /каталог -R`
(см также: http://man7.org/linux/man-pages/man5/core.5.html ; https://linux-audit.com/understand-and-configure-core-dumps-work-on-linux/)

Важно! После вышеуказанных изменений - делаем разрегистрацию и входим в SSH заново.

Вызов краша для проверки настроек

После перерегистрации в SSH проверяем: `ulimit -c` - эта команда теперь должна вывести 'unlimited'.
Затем вызываем искусственный краш какого-либо процесса.

- если ФБ на этом хосте уже установлен, то вводим на консоли команду для принудительного краха ISQL утилиты:

`echo "shell kill -6 $(pgrep isql);" | /opt/firebird/bin/isql -q` Вывод должен быть таким: `Aborted (core dumped)`

Проверяем, что coredump действительно создался:
ls -l /tmp/core.isql.* -rw------- 1 root root 1335296 Oct 5 23:44 /tmp/core.isql.1507227527.2058

Однако, если нет пользовательских подключений, то лучше всего убедиться, что краш будет создаваться также и для СЕРВИСА Firebird (а не только для ПРИЛОЖЕНИЯ, isql, которое сейчас запускалось):

- для Firebird 3.x и старше, работающего в архитектурах `SuperServer` или `SuperClassic`:

```
echo "shell kill -6 $(pgrep firebird);" | /opt/firebird/bin/isql -q
localhost:employee -user sysdba -pas masterkey
```

- для Firebird до 2.5, работающего в архитектуре `SuperClassic`:

```
echo "shell kill -6 $(pgrep fb_smp_server);" | /opt/firebird/bin/isql -q
localhost:employee -user sysdba -pas masterkey
```

- если Firebird работает в архитектуре `Classic`, то сначала надо подключиться к базе, а затем получить PID серверного процесса, обслуживающего это подключение: `select mon$server_pid from mon$attachments where mon$attachment_id = current_connection;`
  Далее ввести `shell kill -6 <найденный_PID>;` - и убедиться после этого, что дамп действительно создался.

- если Firebird на этом хосте отсутствует, то можно вызвать краш какой-либо вспомогательной утилиты, например 'sleep':

```
[root@foo ~]# /bin/sleep 5& kill -s SIGSEGV $(pgrep sleep)
```

Вывод будет примерно таким:
[1] 6773 [root@foo ~]#

-- и здесь, скоре всего, придется еще раз нажать Enter, чтобы в ответ появилось сообщение:

```
[1]+ Segmentation fault (core dumped) /bin/sleep 5
```

Проверяем, что coredump действительно создался:
ls -l /tmp/core.sleep.* -rw------- 1 root root 397312 Oct 6 00:48 /tmp/core.sleep.6773.1507240089

Установка Firebird и отладочного пакета к нему

Устанавливаем Firebird (если его нет на хосте) и, самое главное, добавляем также в его каталог .debug-файлы, которые обязательно должны находиться внутри каталогов с бинарными утилитами/библиотеками, и еще на 1 уровень ниже,
внутри подкаталогов с именами ".debug", т.е.:

```
/opt/firebird/bin/.debug/firebird.debug
/opt/firebird/bin/.debug/gbak.debug
/opt/firebird/bin/.debug/isql.debug
...

/opt/firebird/lib/.debug/libfbclient.so.3.0.5.debug
/opt/firebird/lib/.debug/libib_util.so.debug

/opt/firebird/plugins/.debug/libfbtrace.so.debug
/opt/firebird/plugins/.debug/libEngine12.so.debug
...
```

Внимание! Для каждого снапшота Firebird .debug-файлы отличаются. НЕЛЬЗЯ оставлять старые .debug-файлы при обновлении ФБ-сборки.
Необходимо обновлять весь набор, т.е. переписывая новые .debug'и поверх старых.

Также устанавливаем (при необходимости) LINUX-отладчик gdb: `yum -y install gdb`

Создание дампов при выключенном BugCheckAbort в CentOS-6 (только для Firebird 2.5.x)

Для Firebird 2.5, работающего под управлением CentOS-6, может оказаться актуальной следующая проблема:
дамп не создается до тех пор, пока в firebird.conf не выставлен параметр `BugCheckAbort = 1`. Выставление этого параметра иногда может оказаться неприемлемым, но получение дампов при этом всё-таки возможно.
Чтобы дампы создавались в любом случае, независимо от значения `BugCheckAbort`, необходимо открыть в редакторе скрипт запуска ФБ-службы (`/etc/init.d/firebird`) и добавить строку `export DAEMON_COREFILE_LIMIT=unlimited` перед запуском службы.

Скрипт при этом должен будет выглядеть так:

```
# cat -n /etc/init.d/firebird | tail --lines=+65 | head -30
    65          GUARDIAN=/opt/firebird/bin/fbguard
    66  fi
    67
    68  # initialize as "success"
    69  RETVAL=0
    70
    71  # See how we were called.
    72  case "$1" in
    73    start)
    74          echo -n "Starting $FULLNAME "
    75          export DAEMON_COREFILE_LIMIT=unlimited
    76          daemon --user=$FBRunUser "export FIREBIRD LD_LIBRARY_PATH;
$GUARDIAN -pidfile $pidfile -daemon -forever"
    77          RETVAL=$?
    78          [ $RETVAL -eq 0 ] && touch /var/lock/subsys/$name
    79          echo
    80          ;;
    81    stop)
    82          if [ -f $pidfile ]
    83          then
    84                  echo -n "Stopping $FULLNAME: "
```

Создание трассы

1) создать файл файл gdb-commands.txt - там три строки

```
thread apply all bt
quit
yes
```

2) Отладчиком gdb подцепиться и снять трассу - пример для процесса firebird (т.е. Firebird 3)

```
gdb -q -x ./gdb-commands.txt /opt/firebird/bin/firebird $(pgrep firebird)
1>./stacktrace.txt 2>&1
```

Ручное получение файла дампа

```
gcore -o /путь/префикс $(pgrep firebird)
```

Утилита gcore (входит в пакет gdb) добавит к префиксу идентификатор процесса.

Создание трассы в случае уже существующего дампа (от BugCheckAbort)

```
gdb -q -x ./gdb-commands.txt /opt/firebird/bin/gfix
/tmp/core.gfix.23041.1559703551   1>./gfix.trace1.txt 2>&1
```

Как прибить процесс

```
/opt/firebird/bin/isql -q
shell rm -f /tmp/tmp4crash.tmp ;
create database '/tmp/tmp4crash.tmp' user 'SYSDBA' password 'masterkey' ;
shell kill -6 $(pgrep firebird) ;
```

--- в этот момент мы должны будем вывалиться из isql ---

Вопросы? Пишите на support@ibase.ru.