STM Developer Suite

Программа Spy

Руководство пользователя

Версия 1.0.7.912

Содержание

Программа Spy

Назначение программы

Программа Spy предназначена для:

• автоматического запуска задач после рестарта операционной системы на сервере (например "Master17" и \ или менеджеров потоков "STRM") (далее – контролируемые процессы или процессы);

• запуска задач по команде пользователя после их останова на сервере (например "Master17" и \ или менеджеров потоков "STRM");

• автоматического контроля работоспособности запущенных ранее процессов (и их запуска при необходимости);

• автоматического контроля уникальности в каждый момент времени запущенных ранее процессов (исключения дубликатов);

• остановки запущенных на сервере процессов (например "Master17" и \ или менеджеров потоков "STRM") по команде пользователя.

Системные требования

Программа Spy может быть запущена на IBM-совместимом компьютере под управлением одной из перечисленных операционных систем: Linux, FreeBSD. Для выполнения аналогичных функций под управлением других ОС требуется использование других программных модулей.

Для корректной работы программы необходимы:

• файл настроек программы (spy.conf);

• файлы, содержащие скрипты управления порождаемыми процессами (скрипты для запуска, остановки процессов, напр., *.sh);

• исполняемые (бинарные) файлы, которые должны быть запущены в процессе работы Spy.

Файл настроек программы (spy.conf)

Наличие файла настроек программы является обязательным условием для запуска Spy. В данном файле задаются настройки, в соответствии с которыми программа будет работать. Настройки состоят из 4 секций:

А. Общие опции. Здесь описываются общие настройки программы, а именно:

• LogFile – имя файла, в который будет выводиться отладочная информация и\или информация об этапах работы программы (*.log). Лог файл – это текстовый

файл, с максимальным размером 1 Мбайт; по достижении максимального размера *.log файл удаляется, запись продолжается во вновь созданный файл;

• ShellName - имя командной оболочки, с помощью которой будут запускаться файлы скриптов. В данной версии программы опция не используется, выбор командной оболочки должен осуществляться в скриптах (см. Скрипты запуска \ завершения процессов на стр. 1-9);

• CheckPeriod - период цикла проверки работоспособности контролируемых процессов, в секундах;

• KillScript – имя файла, содержащего скрипт автоматического завершения одного из контролируемых процессов. Имя процесса передается в качестве параметра при запуске скрипта. Скрипт вызывается при закрытии программы **Spy** для каждого процесса.

Б. Перечень процессов. Здесь перечисляются имена контролируемых процессов, каждому имени задается порядковый номер. Имя процесса должно быть идентично имени исполняемого файла, так как оно (имя) участвует при поиске идентификатора процесса в списке запущенных процессов в системе.

Семантика секции:

```
Programs
{
<номер программы>,<имя бинарного файла>;
...
}
```

где:

• номер программы – номер процесса в списке контролируемых процессов, номер 0 – зарезервирован;

• имя бинарного файла – имя процесса (бинарного файла).

В. Запуск процессов. Здесь задается шаблон условий, при выполнении которых запускается скрипт запуска процесса, а так же процедура вызова данного скрипта.

Семантика секции:

```
Actions
{
<список состояний процессов>:
[<список состояний процессов>:]
<имя скрипта>,<таймаут>;
…
}
```

где:

• <список состояний процессов> - здесь под состоянием процесса понимается логическая величина: 0 – не запущен, 1 – запущен. Состояния процессов в списке разделяются запятыми. Над состояниями процессов в одном списке производится операция «логическое И», если результат операции «1» - скрипт выполняется, «0» - не выполняется. Если списков состояний несколько, то над ними выполняется операция «логическое ИЛИ», если результат операции «1» - скрипт выполняется, «0» - не выполняется. Если скрипт не выполнялся, следующая попытка анализа состояний процессов произойдет по истечению указанного таймаута.

Пример: пусть **Spy** контролирует 4 процесса, причем первые три из них должны запускаться независимо от состояний других процессов, а 4-й – при условии успешного запуска первого и второго или второго и третьего. В этом случае списки состояний процессов должны выглядеть так:

```
1/0: 'скрипт_для_запуска_1-го_процесса', 30;
2/0: 'скрипт_для_запуска_2-го_процесса', 30;
3/0: 'скрипт_для_запуска_3-го_процесса', 30;
1/1,2/1,4/0:
2/1,3/1,4/0:
'скрипт_для_запуска_4-го_процесса', 30;
```

• <имя скрипта> - имя файла, содержащего скрипт запуска процесса;

• <таймаут> - таймаут в секундах. Используется между повторами попыток выполнения скрипта запуска процесса при неудаче предыдущей попытки запуска и между повторами проверок необходимости запуска скрипта.

Г. Контроль уникальности процесса в операционной системе. Здесь ставится соответствие между номером процесса и скриптом командной оболочки. Скрипт вызывается автоматически при обнаружении программой нескольких запущенных процессов с одинаковым именем. Имя процесса передается в качестве параметра при запуске скрипта.

Семантика секции:

```
Unique
{
<номер процесса>:<имя скрипта>;
...
}
```

где:

• номер программы – номер процесса в списке контролируемых процессов;

• имя скрипта – имя скрипта.

Общее замечание для всех секций: при задании строковой константы она (кон-

станта) должна быть заключена в одинарные кавычки, например: LogFile='spy.log'.

Пример файла настроек программы:

```
LogFile='spy.log';
ShellName='sh';
CheckPeriod=15;
KillScript = 'killpc.sh';
Programs
{
1,'master17';
2, 'strm';
}
Actions
{
1/0:'m_start.sh',30;
1/1,2/0:'strm_start.sh',30;
}
Unique
{
1:'kill_master.sh';
2:'kill_strm.sh';
}
```

Скрипты запуска/завершения процессов

Скрипты запуска \ завершения процессов пишутся администратором или наладчиком. Требования к содержимому файлов определяются общими требованиями операционной системы. Выбор оболочки и языка, на котором написан скрипт, определяются личными навыками и предпочтениями пользователя.

Для приведенного выше примера файла настройки, приведем примеры использованных в нем скриптов с использованием программной оболочки «sh». В примере все исполняемые файлы находятся в директории /stm, все файлы скриптов (в том числе файл 'functions') находятся в директории /stm/spy.

Примеры содежимого файлов:

Содержимое файла 'killpc.sh':

```
#!/bin/sh
. functions
die "$1"
```

Возможное содержимое файла 'm_start.sh':

```
#!/bin/sh
cd /stm
./master17 test1.stm &
```

Возможное содержимое файла 'strm_start.sh':

```
#!/bin/sh
cd /stm
./strm test1.stm &
```

Возможное содержимое файла 'kill_master.sh':

```
#!/bin/sh
. functions
die "$1"
```

Возможное содержимое файла 'kill_strm.sh':

```
#!/bin/sh
. functions
die "$1"
```

Содержимое файла 'functions':

```
function die() {
          PIDS="`/sbin/pidof $*`"
          while [ ! -z "$PIDS" ]; do
                PIDS="`/sbin/pidof $*`"
                for PID in $PIDS; do
                      count=0
                      while kill $PID >/dev/null 2>&1; do
                            if [ "$count" -gt 60 ]; then
                                  kill -9 $PID
                            fi
                            sleep 1
                            let "count++"
```

```
                    done
                done
            done
}
```

Командная строка

При запуске программы в командной строке должен быть указан полный путь к директории, в которой находится файл настройки программы и файлы скриптов.
Пример:

```
spy /stm/spy
```

Исполняемые файлы должны либо находиться в той же директории, что и исполняемый файл spy, либо пути к ним должны быть прописаны в файлах скриптов.

Контроль процесса работы

Контроль функционирования программы осуществляется с помощью лог файла. В процессе работы, **Spy** выводит в лог файл сообщения об этапах работы, запуске скриптов, результатах выполнения скриптов.

Общий вид строки в лог файле:
[время] [файл и номер строки]> [описание события], например:

```
01.01.07 13:45:00 main.cpp:92> Started /stm/stm237 !
```

Поле «время» - здесь указывается момент времени наступления некоторого события, которое должно быть отражено в лог файле;

Поле «файл и номер строки» - это имя файла с исходным кодом программы и номер строки в нем. Для пользователя эта информация не имеет большого значения.

Поле «описание события» - собственно, именно здесь и находится описание события.

Типы событий, которые отражаются в лог файле:

• Старт программы. Помимо информации о том, что программа запустилась, здесь указывается полученная информация из командной строки.

Пример:

```
Started /stm/stm237 !
```

• Завершение программы. При завершении программы, может быть выведено несколько строк в лог файл. Сначала идет описание причины закрытия программы (а именно код сигнала, полученного от операционной системы), затем строка о завершении программы.

Пример:

```
sig=15
Stopped!
```

• разбор файлов скриптов по старту программы.

Пример:

```
set script <имя_скрипта> to 60
```

• Запуск файла скрипта и результат исполнения скрипта.

Пример:

```
exec <имя_скрипта>
код возврата <код>
```

Завершение программы

Каких-либо определённых требований к закрытию программы нет, по-этому процесс завершения работы **Spy** идентичен процессу закрытия любого приложения для конкретной операционной системы.