Применение Expect в администрировании системы Linux : Часть 1. Функциональные возможности Expect

Алексей Снастин
независимый разработчик ПО
начальник отдела

21.01.2010

Диапазон задач системного администрирования весьма широк и разнообразен. О средствах решения этих задач можно сказать то же самое, в особенности для unix-подобных систем. Об одном из инструментов администрирования – Expect – и пойдёт речь в данном цикле статей. В первой части даётся общее представление об Expect, а также о принципах и механизмах его функционирования. Во второй статье будет описано практическое применение Expect и некоторые аспекты его использования. В третьем, заключительном материале мы рассмотрим работу Expect в сетевой среде.

Что умеет делать Expect

Инструментальное средство Expect предназначено в первую очередь для управления интерактивными программами, которые выводят приглашение и ожидают от пользователя ввода ответа с клавиатуры, например, `passwd, ftp, telnet, rlogin, su` и многие другие. В обычных shell-скриптах такие программы требуют обязательного присутствия человека, вводящего ответы на их запросы. Expect же позволяет автоматизировать диалоговый режим с помощью несложных скриптов на базе языка Tcl, а кроме того, обеспечивает пошаговое управление подпроцессами: можно проверять результат каждой операции, и по результату проверки определять, какие входные данные необходимо предоставить.

Автором программы является Дон Лайбис [Don Libes] из Национального института стандартов и технологий (National Institute of Standards and Technology; NIST). Он не случайно взял за основу Tcl (Tool Command Language – инструментальный командный язык, разработчик Джон Остерхаут [John Ousterhout]), поскольку сам по себе Tcl – это функционально полный язык скриптов, синтаксически простой и удобный в использовании. В нём есть команды создания переменных (set), обработки строк, математические операции, управляющие конструкции (if, while, foreach и т.п.), возможность выполнять внешние программы (команды Unix). Большинство команд задаётся так же, как в оболочке shell, т.е. команда и её аргументы просто разделяются пробелами. Фигурные скобки объединяют элементы в блоки и позволяют записывать инструкции на нескольких строках. В качестве

разделителя команд используется точка с запятой; в конце строки и перед закрывающей фигурной скобкой разделитель не обязателен.

Как работает Expect

Expect запускается в режиме командной строки (в консоли или в эмуляторе терминала в среде X) в соответствии со следующим форматом:

```
expect [-dDinNv] [-c <список_команд>] [ [-[f|b]] <файл_скрипта>] [<аргументы>]
```

Флаг `-d` активизирует вывод некоторой диагностической информации, в основном касающейся выполнения внутренних операций команд, таких как expect и interact.

Флаг `-D` активизирует интерактивный отладчик. Более подробно о работе отладчика можно узнать из документации.

Флаг `-i` переводит Expect в интерактивный режим, в котором ожидается ввод команд с клавиатуры, а не из файла скрипта.

Флаги `-n` и `-N` отключают использование файлов ресурсов. Если существует файл $exp_library/expect.rc, то данные из него считываются автоматически при условии, что не задан флаг `-N`. Сразу после этого автоматически считываются данные из файла ~/.expect.rc при условии, что не задан флаг `-n`.

Флаг `-c` предваряет команду или список команд, которые должны быть выполнены до того, как начнётся выполнение команд скрипта. В списке команды отделяются друг от друга точкой с запятой. Команду или список команд необходимо записывать в кавычках, чтобы защитить их от воздействия командной оболочки.

После флага `-f` записывается имя файла скрипта, из которого будут считываться команды.

По умолчанию весь файл скрипта считывается в память перед выполнением. Но иногда необходимо, чтобы выполнялось построчное чтение, например, для работы с устройством стандартного ввода. Чтобы установить такой режим считывания, воспользуйтесь флагом `-b`.

Из форматной записи очевидно, что флаги `-f` и `-b` являются взаимоисключающими и не обязательными. Иными словами, если встречается только символ "-", то Expect будет считывать команды из стандартного устройства ввода `stdin` (как правило, это клавиатура).

Флаг `-v` выводит номер текущей версии и завершает работу программы.

В конце строки могут быть записаны аргументы, передаваемые в скрипт. Эти аргументы сохраняются в форме списка в переменной `argv`. Переменная `argc` инициализируется числовым значением, соответствующим длине списка `argv`.

Как и для любого скриптового языка в Unix-системах, если в первой строке файла, содержащего команды Expect, записать строку идентификации:

```
#!/usr/bin/expect -f
```

и установить для него статус "выполняемый" в наборе прав доступа, то можно запускать непосредственно этот файл как обычную программу. Разумеется, здесь необходимо указать полный путь к файлу expect в вашей системе.

Набор команд, используемых в Expect

Вообще говоря, для написания Expect-скриптов не требуется глубокое знание языка Tcl, поскольку в самой Expect реализованы дополнительные команды-расширения, обеспечивающие почти всю функциональность. Вот основные, наиболее часто используемые команды:

`spawn` – запуск подпроцесса (в одном скрипте можно запустить несколько подпроцессов для взаимодействия с несколькими внешними программами).

`send` – отправка входных данных в подпроцесс.

`expect` – получение выходных данных из подпроцесса и выполнение соответствующего действия. Эта команда записывается в следующем формате:

```
expect "шаблон" {инструкция ... }
```

Если "шаблон" встречается в строке выходных данных, то выполняется соответствующая инструкция.

`interact` – эта команда позволяет выполнить часть задач в "автоматическом режиме", а затем передать управление пользователю. В свою очередь, пользователь может в любой момент вернуть управление скрипту.

Чтобы лучше понять, как применяются эти команды на практике, рассмотрим простой пример. Приведённый ниже скрипт пересылает файл /etc/passwd с одного компьютера на другой, используя программу ftp.

```
spawn /usr/bin/ftp <сетевое_имя|IP-адрес передающего компьютера>
while 1 { expect {
  "Name*: "   {send "<регистрационное_имя_клиента>\r"}
  "Password:" {send "<пароль>\r"}
  "ftp> "     {break}
  "failed"    {send_user "Нет доступа к FTP-серверу.\r"; exit 1}
  timeout     {send_user "Время ожидания ответа истекло."\r"; exit 2}
}}
send "lcd /etc\r"
expect "ftp> "  {send "cd pub/sysfiles\r"}
expect "ftp> "  {send "get passwd\r"}
expect "ftp> "  {send "quit\r"; send_user "\r"}
exit 0
```

Данный скрипт запускается на принимающем компьютере. В первой строке инициализируется подпроцесс интерактивного ftp-клиента с указанием сетевого имени или

IP-адреса компьютера, выполняющего роль сервера, – с него будет передаваться требуемый файл.

Цикл while языка Tcl позволяет организовать регистрацию клиента на ftp-сервере, которая в обычных условиях представляет собой диалоговую процедуру. Здесь же expect ожидает ответ сервера, сравнивает его с заданными шаблонами и, если один из этих шаблонов найден в строке ответа сервера, то выполняет соответствующую команду или команды. Обычно сервер начинает процедуру регистрации с запроса имени (первый шаблон в блоке expect – в ответ отсылается имя ftp-клиента) и пароля (второй шаблон). Если имя и пароль указаны правильно, то процедура регистрации считается успешно завершённой, и сервер выдаёт приглашение "ftp> " (третий шаблон). В этом случае происходит выход из цикла (команда break). Последние два шаблона соответствуют исключительным ситуациям. По тем или иным причинам сервер может запретить регистрацию клиента с заданным именем и паролем и выдать сообщение "Login failed". При обнаружении шаблона "failed" в строке ответа сервера пользователю-клиенту выдаётся сообщение о недоступности сервера (ещё одна внутренняя команда программы Expect – `send_user` – строка не передаётся в подпроцесс, как в команде send, а записывается в стандартный поток вывода), после чего скрипт завершает своё выполнение с кодом ошибки (exit 1). Ключевое слово timeout позволяет реагировать на ситуации, когда в течение 10 секунд ничего не происходит (десятисекундный тайм-аут принят по умолчанию в программе Expect; вы можете изменить это значение, например, "set timeout 30"). В нашем случае выводится сообщение об истечении времени ожидания, и скрипт завершается с кодом ошибки exit 2.

Далее записана последовательность команд, отправляемых ftp-серверу. Здесь необходимо отметить, что каждая команда expect ожидает завершения выполнения предыдущей команды (выдачи приглашения "ftp> "). После того как файл passwd скопирован, закрывается сеанс работы с ftp-сервером (посылается команда "`quit`"), и скрипт заканчивает свою работу с кодом успешного выполнения (`exit 0`).

Заключение

В статье мы описали лишь некоторые внутренние команды программы Expect и простейший пример скрипта, демонстрирующий основные приёмы использования этой программы. Но даже на этом кратком примере видны несомненные преимущества использования Expect для задач системного администрирования.

В следующих статьях цикла будут рассмотрены скрипты, действительно используемые в практической работе системных и сетевых администраторов.

Об авторе

Алексей Снастин

Алексей Снастин - независимый разработчик ПО, консультант и переводчик с английского языка технической и учебной литературы по ИТ. Принимал участие в разработке сетевых офисных приложений типа клиент/сервер на языке C в среде Linux.