**Глава 19. Подоболочки, или Subshells**

Запуск сценария приводит к запуску дочернего командного интерпретатора. Который выполняет интерпретацию и исполнение списка команд, содержащихся в файле сценария, точно так же, как если бы они были введены из командной строки. Любой сценарий запускается как дочерний процесс [родительской](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c5358.html#FORKREF) командной оболочки, той самой, которая выводит перед вами строку приглашения к вводу на консоли или в окне xterm.

Сценарий может, так же, запустить другой дочерний процесс, в своей подоболочке. Это позволяет сценариям распараллелить процесс обработки данных по нескольким задачам, исполняемым одновременно.

**Список команд в круглых скобках**

( command1; command2; command3; ... )

Список команд, в круглых скобках, исполняется в подоболочке.

|  |  |
| --- | --- |
| Note | Значения переменных, определенных в дочерней оболочке, *не* могут быть переданы родительской оболочке. Они недоступны [родительскому процессу](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c5358.html#FORKREF). Фактически, они ведут себя как [локальные переменные](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x12644.html). |

**Пример 19-1. Область видимости переменных**

#!/bin/bash

# subshell.sh

echo

outer\_variable=Outer

(

inner\_variable=Inner

echo "Дочерний процесс, \"inner\_variable\" = $inner\_variable"

echo "Дочерний процесс, \"outer\" = $outer\_variable"

)

echo

if [ -z "$inner\_variable" ]

then

echo "Переменная inner\_variable не определена в родительской оболочке"

else

echo "Переменная inner\_variable определена в родительской оболочке"

fi

echo "Родительский процесс, \"inner\_variable\" = $inner\_variable"

# Переменная $inner\_variable не будет определена

# потому, что переменные, определенные в дочернем процессе,

# ведут себя как "локальные переменные".

echo

exit 0

См. также [Пример 31-1](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c13371.html#SUBPIT).

+

Смена текущего каталога в дочернем процессе (подоболочке) не влечет за собой смену текущего каталога в родительской оболочке.

**Пример 19-2. Личные настройки пользователей**

#!/bin/bash

# allprofs.sh: вывод личных настроек (profiles) всех пользователей

# Автор: Heiner Steven

# С некоторыми изменениями, внесенными автором документа.

FILE=.bashrc # Файл настроек пользователя,

#+ в оригинальном сценарии называется ".profile".

for home in `awk -F: '{print $6}' /etc/passwd`

do

[ -d "$home" ] || continue # Перейти к следующей итерации, если нет домашнего каталога.

[ -r "$home" ] || continue # Перейти к следующей итерации, если не доступен для чтения.

(cd $home; [ -e $FILE ] && less $FILE)

done

# По завершении сценария -- нет теобходимости выполнять команду 'cd', чтобы вернуться в первоначальный каталог,

#+ поскольку 'cd $home' выполняется в подоболочке.

exit 0

Подоболочка может использоваться для задания "специфического окружения" для группы команд.

COMMAND1

COMMAND2

COMMAND3

(

IFS=:

PATH=/bin

unset TERMINFO

set -C

shift 5

COMMAND4

COMMAND5

exit 3 # Выход только из подоболочки.

)

# Изменение переменных окружения не коснется родительской оболочки.

COMMAND6

COMMAND7

Как вариант использования подоболочки -- проверка переменных.

if (set -u; : $variable) 2> /dev/null

then

echo "Переменная определена."

fi

# Можно сделать то же самое по другому: [[ ${variable-x} != x || ${variable-y} != y ]]

# или [[ ${variable-x} != x$variable ]]

# или [[ ${variable+x} = x ]])

Еще одно применение -- проверка файлов блокировки:

if (set -C; : > lock\_file) 2> /dev/null

then

echo "Этот сценарий уже запущен другим пользователем."

exit 65

fi

# Спасибо S.C.

Процессы в подоболочках могут исполняться параллельно. Это позволяет разбить сложную задачу на несколько простых подзадач, выполняющих параллельную обработку информации.

**Пример 19-3. Запуск нескольких процессов в подоболочках**

(cat list1 list2 list3 | sort | uniq > list123) &

(cat list4 list5 list6 | sort | uniq > list456) &

# Слияние и сортировка двух списков производится одновременно.

# Запуск в фоне гарантирует параллельное исполнение.

#

# Тот же эффект дает

# cat list1 list2 list3 | sort | uniq > list123 &

# cat list4 list5 list6 | sort | uniq > list456 &

wait # Ожидание завершения работы подоболочек.

diff list123 list456

Перенаправление ввода/вывода в/из подоболочки производится оператором построения конвейера "|", например, **ls -al | (command)**.

|  |  |
| --- | --- |
| Note | Блок команд, заключенный в *фигурные скобки* не приводит к запуску дочерней подоболочки.  { command1; command2; command3; ... } |

|  |  |  |
| --- | --- | --- |
| [Назад](http://www.opennet.ru/docs/RUS/bash_scripting_guide/x12282.html) | [К началу](http://www.opennet.ru/docs/RUS/bash_scripting_guide/index.html) | [Вперед](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c12376.html) |
| Globbing -- Подстановка имен файлов | [Наверх](http://www.opennet.ru/docs/RUS/bash_scripting_guide/p11889.html) | Ограниченный режим командной оболочки |