

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** 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, выполняется в подоболочке.



Значения переменных, определенных в дочерней оболочке, *не* могут быть переданы родительской оболочке. Они недоступны **родительскому процессу**. Фактически, они ведут себя как **локальные переменные**.

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](#).

+

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 необходимости 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 подболочек.

diff list123 list456

```

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



Блок команд, заключенный в *фигурные скобки* не приводит к запуску дочерней подболочки.

```
{ command1; command2; command3; ... }
```

