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, исполняется 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" ]
  echo "Переменная inner variable не определена в родительской
оболочке"
  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
```

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
# Abrop: Heiner Steven
# С некоторыми изменениями, внесенными автором документа.
FILE=.bashrc # Файл настроек пользователя,
              #+ в оригинальном сценарии называется ".profile".
for home in `awk -F: '{print $6}' /etc/passwd`
  [ -d "$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
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.

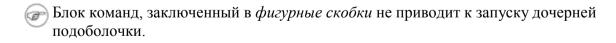
```
if (set -u; : $variable) 2> /dev/null
  есho "Переменная определена."
fi
# Можно сделать то же самое по другому: [[ \{variable-x\} != x | \}
${variable-v} != v ]]
# или
                                          [[ ${variable-x} != x$variable
11
# или
                                          [[ \{\text{variable}+x\} = x ]])
Еще одно применение -- проверка файлов блокировки:
if (set -C; : > lock file) 2> /dev/null
 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, 1s -al | (command).



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