# SH(1)

## ИМЯ

sh - Вызывает командный интерпретатор shell.

## СИНТАКСИС

```
sh [-ceiknrstuvx] [args]
```

## ОПИСАНИЕ

Shell - это стандартный командный язык программирования, который выполняет чтение команд с терминала или из файла. Ниже смотрите описание раздела ВЫЗОВ (INVOCATION), в котором описаны значения аргументов интерпретатора shell.

## КОМАНДЫ

Простая команда(simple-command) - это последовательность слов(words), не содержащих пустых символов и разделенных пробелами (пустой символ(blank) - это знак табуляции или пробел). Первое слово определяет имя выполняемой команды. За исключением того что определено ниже, оставшиеся слова передаются как аргументы в вызванную команду. Имя команды передается как аргумент 0 (смотрите команду exec(2)). В случае нормального завершения работы простой команды значение (value) - это состояние ее выхода, а в случае аварийного завершения (восьмеричное значение)1000+ значение состояния (status) (т.е. если аварийно завершилась основная процедура файла). Смотрите команду signal(S), где приведен список оценок состояний.

Конвейер(pipeline) - это последовательность одной или более команд, разделенных символом вертикальная черта (|). (Символ ^ устаревший синоним для символа вертикальная черта и его не следует использовать в конвейере).Стандартный вывод каждой команды, кроме последней, связан каналом pipe(S) со стандартным вводом следующей команды. Каждая команда выполняется как отдельный процесс; интерпретатор shell ждет завершения последней команды.

Список(list) - это последовательность одного или более конвейеров, разделенных символами ;,&,&& или ||(две вертикальные черты) и необязательно завершающихся символами ; или &. Из этих четырех символов, символы ; и & имеют одинаковый приоритет, который ниже чем у символов && и ||. Символы && и || также имеют равный приоритет. Точка с запятой (;) вызывает последовательное выполнение предыдущего конвейера; амперсанд (&) вызывает асинхронное выполнение предыдущего конвейера (т.е. интерпретатор shell не ждет завершения этого конвейера). Символ &&(||) вызывает список(list) следующий за ним для выполнения, только если состояние выхода предыдущего конвейера равно нулю (не ноль). Произвольное количество символов новых строк может появиться в списке (list) вместо точки с запятой (;) для разделения команд.

Команда (a command) представляет собой либо простую команду, либо одну из перечисленных ниже. Значение,возвращаемое командой,определяется значением последней простой команды выполненной в команде, кроме значений установленных иначе.

```
for name [in word...]

do

list
```

```
done
```

Каждый раз при выполнении команды for аргумент name устанавливается для следующего слова(word), взятого из списка in word. Если аргумент in word... отсутствует, то команда for выполнит аргумент do list один раз для каждого установленного позиционного параметра (смотрите ниже описание раздела ПОДСТАНОВКА ПАРАМЕТРА (Parameter Substitution)). Выполнение завершается, когда в списке больше нет слов(words).

```
case word in

[ pattern [|pattern ]...)list

 ;;]

case
```

Команда case выполняет список list, связанный с первым шаблоном(pattern), который соответствует аргументу word. Форма шаблонов совпадает с формой, используемой при генерации имени файла (смотрите описание раздела ГЕНЕРАЦИЯ ИМЕНИ ФАЙЛА (Filename Generation)).

```
if list then

list

[ elif list then

list ]

[ else list ]

fi
```

Аргумент list,следующий за командой if выполняется и, если возвращаемое состояние выхода равно 0, то выполняется список list следующий за первым параметром then. Иначе, выполнится список list, следующий за elif и, если его значение равно 0, то выполнится список list, относящийся к следующему then. В противном случае, выполнится else list. Если не выполняются аргументы else list или then list, то команда if возвращает нулевое состояние выхода.

```
while list

do

list

done
```

Команда while неоднократно выполняет параметр while list и, если состояние выхода последней команды в списке равно нулю, то выполняет do list; в противном случае, цикл завершается. Если команды в аргументе do list не выполняются, то команда while возвращает нулевое состояние выхода; команда until может быть использована вместо команды while для отмены проверки завершения цикла.

(list)
: Выполнять аргумент list в субинтерпретаторе sub-shell

{ list;}
: Аргумент list просто выполняется

name () { list;}
Определить функцию, на которую ссылаются с помощью аргумента name. Тело функций это список (list) команд между фигурными скобками {}. Ниже описано выполнение функций (смотрите описание раздела ВЫПОЛНЕНИЕ (Execution)).

Следующие слова определяются только как первое слово команды, когда не взяты в кавычки:

```
if then else elif fi case esac for while until do done { }
```

## КОММЕНТАРИИ

Слово, начинающееся с символа # означает, что слово и все последующие символы до символа новая строка будут игнорироваться.

## ПОДСТАНОВКА КОМАНДЫ

Стандартный вывод из команды,заключенной в пару штрихов (` `), может быть использован как для части,так и для всего слова; последующие символы новых строк удаляются.

## ПОДСТАНОВКА ПАРАМЕТРА

Символ $ используется для ввода подставляемых параметров(parameters). Позиционные параметры могут быть установлены значениями команды set. Значения переменным могут быть назначены следующей записью:

```
name=value[ name=value]... (имя-значение)
```

Сравнение шаблона с численным значением value не производится.

${parameter}
Параметр(parameter) - это последовательность букв,цифр или символов подчеркивания, имя (name) это цифра или любой из символов *,@,#,?,-,$ и !. Значение параметра,если такое существует, замещается. Фигурные скобки требуются только когда за параметром (parameter) следует буква, цифра или знак подчеркивания, которые не будут интерпретированы как часть его имени. Имя (name) должно начинаться с буквы или символа подчеркивания.Если параметр (parameter) это цифра, то это позиционный параметр. Если параметр (parameter) это символ * или @, то все позиционные параметры, начинающиеся с $1, замещаются (разделенные пробелами). Параметр $0 устанавливается с аргумента ноль,когда происходит вызов shell.

${parameter:-word}
Если параметр (parameter) установлен и не равен нулю, то происходит подстановка его значения, иначе подставляется word.

${parameter:=word}
Если параметр (parameter) не назначен или равен нулю, то установить его в word; значение параметра затем подставляется. Позиционные параметры не могут быть назначены этим способом.

${parameter:?word}
Если параметр (parameter) установлен и не ноль, то происходит подстановка его значения, иначе печатается аргумент word и осуществляется выход из shell. Если аргумент word отсутствует, то печатается сообщение "Параметр равен нулю или не установлен".

${parameter:+word}
Если параметр (parameter) установлен и не равен нулю, то подставить аргумент word; иначе ничего не подставляется. В выше приведенном описании, аргумент word не рассматривается кроме тех случаев, когда он должен быть использован как подставляемая строка, так что в следующем примере команда pwd выполнится если только аргумент d не установлен или равен нулю:

```
echo ${d:-`pwd`}
```

Если двоеточие (:) опушено в вышеприведенном примере, то shell только проверит установлен параметр (parameter) или нет.

Следующие параметры автоматически устанавливаются интерпретатором shell:

**#** количество позиционных параметров в десятичном виде;

**-** состояние флагов,установленных при вызове shell или командой set;

**?** десятичное значение кода возврата последней синхронно выполненной команды;

**$** номер процесса текущего shell;

**!** номер процесса вызвавшего последнюю фоновую команду.

Следующие параметры используются интерпретатором shell:

CDPATH
Путь поиска для команды cd. Смотрите раздел ВСТРОЕННЫЕ КОМАНДЫ (Special Commands) команда "cd".

HOME
Аргумент по умолчанию (входной каталог) для команды cd.

PATH
Путь поиска команд ( смотрите ниже описание раздела ВЫПОЛНЕНИЕ (Execution)).

MAIL
Если эта переменная определяет имя почтового файла, то интерпретатор shell информирует пользователя о прибытии почты в указанном файле.

MAILCHECK
Этот параметр определяет как часто (в секундах) интерпретатор shell будет контролировать прибытие почты в файлах, определенных параметрами MAIL и MAILPATH. Значение по умолчанию 600 секунд(10 минут). Если установлен ноль, то shell будет проверять прибытие почты перед каждым приглашением.

MAILPATH
Двоеточие разделяет список имен файлов. Если этот параметр установлен, интерпретатор shell информирует пользователя о прибытии почты в любом из указанных файлов. За каждым именем файла может следовать символ % и сообщение, которое будет напечатано когда измениться время модификации. Сообщение по умолчанию "you have mail - для вас есть почта".

PS1
Первичная строка подсказка по умолчанию "$".

PS2
Вторичная строка подсказка по умолчанию ">".

IFS
Внутренние разделители полей, обычно пробел, знак табуляции и символ новой строки.

SHACCT
Если этот параметр установлен для имени файла перезаписанного пользователем, то интерпретатор shell будет писать запись учета в файл для каждой выполненной процедуры shell. Программы учета такие как acctcom(8) и accton(8) могут быть использованы для анализа собранных данных. Это свойство используется не во всех версиях интерпретатора shell.

SHELL
При своей активизации,shell просматривает среду в поисках этого имени (смотрите ниже описание раздела СРЕДА (Environment)). Если оно найдено и содержит 'r', как часть его значения, то shell становится ограниченным.

Интерпретатор shell получает по умолчанию следующие значения PATH, PS1, PS2 и IFS, в то время как совсем не устанавливаются интерпретатор shell параметры HOME и MAIL (хотя HOME устанавливается командой login(5)).

### ИНТЕРПРЕТАЦИЯ ПУСТОГО СИМВОЛА

После подстановки параметра и команды, результаты подстановки просматриваются для определения символов внутренних разделителей полей (символы определены в параметре IFS) и разделяются на отдельные аргументы, если таковые символы найдены. Явные нулевые аргументы (" " или ' ') сохраняются. Неявные нулевые аргументы ( возникающие из параметров, которые не имеют значений) удаляются.

### ГЕНЕРАЦИЯ ИМЕНИ ФАЙЛА

Вслед за подстановкой, каждая команда word просматривается для обнаружения символов *,?, и [. Если один из этих символов присутствует, то команда word рассматривается как шаблон (pattern). Команда word замещается именами файлов, отсортированными в алфавитном порядке, которые соответствуют шаблону. Если нет имени файла соответствующего шаблону, то команда word остается неизменной. Символ . в начале имени файла или непосредственно следующий за символом /, так же как сам символ /, должен быть точно указан. Символы и соответствующие им шаблоны:

***** Соответствует любой строке, включая нулевую строку.

**?** Соответствует любому одиночному символу

**[...]** Соответствует любым символам из заданного набора. Пара символов разделенных - (тире) соответствует любому лексическому символу, содержащемуся между парой символов. Если первый символ, следующий за открывающейся квадратной скобкой "[" это символ "!", то это соответствует любому символу не включенному в скобки.

### ЗАКЛЮЧЕНИЕ В КАВЫЧКИ (Quoting)

Следующие символы имеют специальное назначение в интерпретаторе shell и вызывают завершение команды word (кроме случаев, когда эти символы заключены в кавычки):

; & ( ) | ^ < > символ новой строки, пробел, символ табуляции

Символ может быть заключен в кавычки (т.е.подготовлен для установки самого себя) предшествующим ему символом \. Комбинация пары символов \ и знак новой строки игнорируются. Все символы, заключенные между парой одинарных кавычек (' '), за исключением одинарной кавычки(апостроф), берутся в кавычки. Внутри двойных кавычек (" ") осуществляется подстановка параметра и команды и символ \ заключает в кавычки символы \,`," и $. Символ "$ *" эквивалентен "$1 $2...", тогда как символ "$@" эквивалентен "$1" "$2"....

### ПОДСКАЗКА

Когда вы работаете в интерактивном режиме, то перед чтением команды интерпретатор shell выводит подсказку, значение параметра PS1. Если в любой момент времени набрана новая строка и требуется продолжение ввода текста для завершения набора команды, то появляется вторичная подсказка(т.е.значение параметра PS2).

### ПРОГРАММА ПРОВЕРКИ ОРФОГРАФИИ

Когда используется команда cd(1) интерпретатор shell проверяет орфографию введенной команды. Для примера, если вы переходите в другой каталог, используя команду cd и сделали орфографическую ошибку в имени каталога, интерпретатор shell сообщает альтернативную орфографию существующего каталога. Введите "y" и нажмите RETURN (или просто нажмите RETURN) для перехода в предложенный каталог. Если предложенная орфография некорректна, введите "n", затем введите корректную командную строку. В следующем примере, реакция интерпретатора shell на ввод команды sh(1) показана жирным шрифтом:

```
$cd/usr/spol/uucp

cd/usr/spool/uucp?y

ok
```

**ВВОД/ВЫВОД**

Перед выполнением команды, ее ввод и вывод может быть переадресован с помощью специальной системы обозначений интерпретируемой shell. Обозначения могут появится где-либо в простой команде, а также могут предшествовать или следовать за командой (command). Они не переходят в вызванную команду; подстановка располагается перед словом word или цифрой digit:

<word
: Стандартный ввод берется из файла word( дескриптор файла 0).

>word
: Стандартный вывод направляется в файл word( дескриптор файла 1). Если файл не существует, то он создается; иначе, он урезается до нулевой длины.

>>word
: Стандартный вывод направляется в файл word. Если файл существует, то вывод добавляется к нему (путем установки указателя на конец файла); в противном случае, файл создается.

<<[-]word
: Интерпретатор shell считывает ввод до строки, имеющей то же самое значение, что и word, или до конца файла. Результирующий документ становится стандартным вводом. Если любой символ аргумента word заключен в кавычки, то символы документа не интерпретируются, иначе, осуществляется подстановка параметра и команды, (неустраненные) символы \ и знак новой строки игнорируются, а также символ \ должен быть использован для заключения в кавычки символов \,$,' и первого символа аргумента word. Если символ - (черта) добавлен к символам <<, то все впереди стоящие символы табуляции отсекаются из аргумента word и из документа.

<&digit
: Дескриптор файла стандартного ввода получается в результате дублирования дескриптора файла digit. (Смотрите описание команды dup(S)). Подобно этому для стандартного вывода используется конструкция >&digit.

<&-
: Стандартный ввод закрыть. Подобно этому, для стандартного вывода используется комбинация >&-.

Если любой, из выше перечисленных, конструкций предшествует цифра, то файл будет иметь дескриптор, задаваемый этой цифрой (вместо используемых по умолчанию дескрипторов 0 и 1). Например,

```
...2>&1
```

команда создает дескриптор файла 2, который является дублем дескриптора файла 1.

Если за командой следует символ &, то стандартный ввод по умолчанию для команды это пустой файл /dev/null. Иначе,среда для выполнения команды содержит дескрипторы файлов вызываемого shell, измененные спецификациями ввода/вывода.

**СРЕДА**

Среда (environment)(смотрите описание команды environ(5)) это список пар "имя-значение", который передается в выполняемую программу точно также как и обычный список аргументов. Интерпретатор shell

взаимодействует со средой несколькими способами. При вызове, shell просматривает среду и создает параметр для каждого найденного имени, назначая ему соответствующее значение. Выполненные команды наследуют ту же самую среду. Если пользователь изменяет значения этих параметров или создает новые, то никакие из этих параметров не влияют на среду, кроме команды export, которая используется для связи параметров shell со средой. Среда просмотренная любой выполненной командой состоит из любых неизмененных пар "имя-значение" первоначально унаследованных интерпретатором shell, минус любые пары удаленные с помощью команды unset, плюс любые изменения или дополнения, каждые из которых должны быть записаны в командах export.

Среда для любой простой команды (simple-command) может быть увеличена с помощью использования перед ней одного или более назначений параметров. Таким образом:

```
TERM=450cmd args
```

и

```
(export TERM; TERM=450; cmd args)
```

эквивалентны (что касается выполнения выше приведенной команды cmd).

Если флаг -k установлен, то все ключевые аргументы размещаются в среде, даже если они указаны после имени команды.

### СИГНАЛЫ

Сигналы INTERRUPT(прерывание) и QUIT(выход) для вызванной команды игнорируются, если за командой следует символ &; иначе, сигналы имеют значения унаследованные интерпретатором shell от его родительского процесса, с ожиданием сигнала 11. Ниже смотрите описание команды trap).

### ВЫПОЛНЕНИЕ

Каждый раз когда выполняется команда осуществляются вышеприведенные подстановки. Если имя команды не соответствует одной из ВСТРОЕННЫХ КОМАНД, но соответствует имени определенной функции, то функция выполняется в процессе shell (заметим как это отличается от выполнения процедур shell). Позиционные параметры $1,$2... устанавливаются для аргументов функции. Если имя команды не соответствует ни встроенной команде, ни имени определенной функции, то создается новый процесс и осуществляется попытка выполнить команду с помощью команды exec(2).

Параметр shell PATH определяет путь поиска каталога, содержащего команду. Имена альтернативных каталогов разделяются двоеточием(:). Путь по умолчанию это :/bin:/usr/bin (определяя текущий каталог в таком порядке /bin и /usr/bin). Заметим, что текущий каталог определяется именем нулевого пути, которое может появиться сразу после знака равенства или между символами двоеточие(:) в любом месте в списке пути. Если имя команды содержит символ /, то путь поиска не используется. Иначе, каждый каталог указанный в пути поиска просматривается для поиска выполняемого файла. Если файл выполняемый, но не является файлом a.out, то он будет представлять собой файл, содержащий команды shell. Субинтерпретатор sub-shell (т.е.отдельный процесс) создается для его чтения. Заключенная в скобки команда также выполняется в субинтерпретаторе sub-shell.

Местоположение в пути поиска, где была найдена команда, запоминается shell (чтобы избежать лишних действий позднее). Если команда была найдена в связном каталоге, ее местоположение должно быть переопределено всякий раз, когда изменяется текущий каталог. Интерпретатор shell забывает все сохраненные местоположения, всякий раз, когда переменная PATH изменяется или выполняется команда hash-r (описание ее смотрите ниже).

### ВСТРОЕННЫЕ КОМАНДЫ

Для этих команд разрешена переадресация ввода/вывода.

:
Пустая команда. Это команда ничего не делает.Возвращает нулевое состояние выхода.

.file
Считать и выполнить команду из указанного файла(file), после чего вернуться назад. Для поиска каталога, содержащего файл(file), используется путь поиска определенный параметром PATH.

break[ n ]
Выход из объемлющего цикла for или while, если таковой существует. Если задано n, то выполняется выход из уровня n.

continue[ n ]
Начать следующую итерацию объемлющего цикла for или while. Если задано n, то возобновляется выполнение с n-го объемлющего цикла.

cd[ arg ]
Команда изменяет текущий каталог на указанную в аргументе arg. Параметр shell HOME это аргумент arg по умолчанию. Параметр shell CDPATH определяет путь поиска каталога, содержащего arg. Альтернативные имена каталогов разделяются двоеточием. Путь по умолчанию это нулевой (определяющий текущий каталог). Заметим, что текущий каталог определяется именем нулевого пути, которое может появиться сразу после знака равенства или между символами двоеточие(:), в любом месте в списке пути. Если аргумент arg начинается с символа /, то путь поиска не используется. Иначе, каждый каталог из пути просматривается для поиска arg.

Если shell прочитал его команды с терминала и указанного каталога не существует (или не найдены некоторые компоненты), используется исправление орфографии каждой компоненты каталога (directory) для поиска "корректного" имени, затем shell запрашивает надо ли изменять каталог на исправленное имя каталога, ответ "n" подразумевает "нет", любой другой ответ означает "да".

echo[arg...]
Записывает аргументы, разделенные пробелами и заканчивающиеся символом новой строки, в стандартный вывод. Аргументы могут быть заключены в кавычки. Кавычки требуются для того чтобы интерпретатор shell корректно интерпретировал эти специальные escape последовательности:

**\b** возврат на одну позицию

**\c** печатает строку без символа новой строки

**\f** прогон страницы

**\n** символ новой строки

**\r** возврат каретки

**\t** табуляция

**\v** вертикальная табуляция

**** символ backslash

**\n** 8-битовый символ, чей код ASCII представляет собой 1,2,3-х разрядное число в восьмеричной форме, число n должно начинаться с нуля

eval[ arg...]
Аргументы arg читаются интерпретатором shell в качестве ввода, и полученная в результате команда (команды) выполняется.

exec[ arg...]
Вместо интерпретатора shell выполняется команда, задаваемая аргументами, новый процесс не создается. В команде могут присутствовать аргументы ввода/вывода, если кроме них в команде exec нет других аргументы, происходит переназначение ввода/вывода shell.

exit[ n ]
Вызывает завершение работы shell с состоянием выхода n. Если параметр n опущен, состояние выхода будет состояние выхода

последней выполненной команды. Признак конца файла также приведет к выходу из интерпретатора shell.

export[ name...]
Указанные переменные names помечаются для автоматического экспорта в среду (environment) выполняемых далее команд. Если аргументы не заданы, печатается список всех имен экспортируемых в этот shell.

hash[ -r ][ name...]
Для каждого имени (name), местоположение в пути поиска команды указанной параметром name определяется и запоминается интерпретатором shell. Опция -r вызывает shell для устранения всех запомненных местоположений. Если параметры не заданы, то предоставляется информация о запомненных командах. Hits (справки) это количество вызовов команды процессом shell. Cost (затраты) это мера работы, требуемая чтобы разместить команду в пути поиска. Существуют определенные ситуации, в которых требуется, чтобы сохраненные размещения команды были пересчитаны. Команды, для которых это будет сделано помечаются звездочкой(*), стоящей рядом с информацией hits. Cost будет наращена, после того как сделают пересчет.

newgrp[ arg...]
Эквивалентна команде exec newgrp arg...

pwd Печатает текущий работающий каталог.Смотрите pwd(1) для ознакомления с описанием и использованием команды.

read[ name...]
Из стандартного ввода считывается одна строка и первое слово присваивается первой переменной name, второе слово второй переменной name и т.д., оставшиеся слова присваиваются последней переменной name. Ненулевое состояние выхода возвращается только при достижении конца файла.

readonly[ name...]
Полученные переменные names помечаются как доступные только для чтения (readonly), значения этих переменных не могут быть изменены с помощью последующего назначения. Если параметры не заданы, печатается список всех переменных, доступных только для чтения (readonly).

return[ n ]
Вызывает функцию для выхода с значением возврата, указанным в n. Если n опущено, то состояние возврата соответствует состоянию последней выполненной команды.

set[ -eknuvx[ arg...]]

**-e** Если интерпретатор shell неинтерактивный,то осуществляется немедленный выход, в случае если команда завершилась с ненулевым кодом завершения работы.

**-f** Запрет формирования имени файла.

**-h** Разместить и запомнить функциональные команды как функции,которые определены (функциональные команды обычно размещаются, когда функция выполнена).

**-k** Поместить в среду все ключевые параметры для команды, а не только те, которые предшествуют имени команды.

**-n** Считать команды, но не выполнять их.

**-u** Считать ошибкой подстановку неустановленных переменных.

**-v** Печатать входные строки shell по мере их считывания.

**-x** Печатать команды и их параметры по мере их выполнения. Хотя этот флаг передается в субинтепретатор (sub-shell), он не может быть отслежен в этих субинтерпретаторах.

**--** Не изменять любые флаги; полезен в установке $1 в -.

Для отмены назначений флагов лучше использовать символ + вместо символа -. Эти флаги также могут задаваться при вызове интерпретатора shell. Текущую установку флагов можно найти в параметре $-. Оставшиеся аргументы это позиционные параметры и

присваиваются по порядку переменным $1,$2,... Если не задано ни одного параметра, то будут распечатаны значения всех переменных.

shift[n]
Позиционные параметры с $2... переименовываются в $1...

test Вычисляет условные выражения. Смотрите test(1) для ознакомления с описанием и использованием команды.

times
Выдать суммарные времена - время пользователя и системное время, затраченные на выполнение процессов, запущенных из интепретатора shell.

trap[ arg ][ n ]...
Параметр arg представляет собой команду, которую следует считать и выполнить при получении сигнала (сигналов) n.(Заметим,что параметр вычисляется дважды: сначала при установке прерывания на сигнал, а потом при его обработке). Команды прерывания trap выполняются в порядке номера сигнала. Максимальный номер сигнала - 16. Любые попытки к установке прерывания на сигнал, который был игнорирован на входе в текущий shell не эффективны. Попытка установить прерывание на сигнал 11(дефект памяти) приведет к ошибке. Если параметр arg отсутствует, то все прерывания n восстанавливаются в их исходные значения. Если параметр arg является пустой строкой, то указанный сигнал игнорируется интерпретатором shell и вызываемыми из него командами. Если n равно нулю, то параметр arg команды выполнится при выходе из shell. Команда trap без параметров печатает список команд связанных с каждым номером сигнала.

type[ name...]
Каждое имя (name) помечается как оно будет интерпретироваться, если использовано как имя команды.

ulimit[[ -f ]n ]
Налагает ограничение размера блоков n в файлах.

**-f** налагает ограничение размера блоков n в файлах, созданных процессами-потомками (могут быть прочитаны файлы любого размера). Любой пользователь может уменьшить ограничение размера файла, но увеличить ограничение может только привилегированный пользователь, находящийся в корневом каталоге. Если команда без аргументов, то будет напечатано текущее ограничение. Если опции не заданы, то назначается опция -f.

unset[ name...]
Для каждого имени (name), команда удаляет соответствующую переменную или функцию. Не могут быть использованы следующие переменные PATH,PS1,PS2,MAILCHECK и IFS, так как их нельзя сбрасывать.

umask[ ooo ]
Маска пользователя при создании файла устанавливается в восьмеричный номер ooo, где o - восьмеричная цифра (смотрите umask(1)). Если ooo опущено, то будет напечатано текущее значение маски.

wait[ n ]
Эта команда ждет окончания работы заданного процесса и сообщает его состояние завершения. Если n не задано, то команда wait ждет завершения всех процессов-потомков, активных в данный момент. Код возврата этой команды всегда равен нулю.

## ВЫЗОВ

Если shell вызван через команду exec(2) и первый символ нулевого параметра это -, то команды первоначально читаются из входного файла /etc/profile, а затем из файла $HOME/.profile, если такие файлы существует. После этого, команды считываются как описано ниже, что происходит также в случае, когда shell вызван как /bin/sh. Приведенные ниже флаги интерпретируются shell только при вызове. Отметим, что

кроме того что указывается флаг -c или -s, первому аргументу назначается имя файла содержащего команды, и оставшиеся аргументы передаются как позиционные параметры в этот командный файл:

**-c string** Если задан флаг -c, то команда считывается из строки string.

**-s** Если задан флаг -s или если аргументы не сохраняются, то команды считываются из стандартного ввода. Любые оставшиеся аргументы определяют позиционные параметры. Вывод интерпретатора shell записывается в файл с дескриптором 2.

**-t** Если задан флаг -t, то читается и выполняется одиночная команда и интерпретатор shell завершается. Этот флаг предназначается для использования только программами написанными на языке C и бесполезен в интерактивном режиме.

**-i** если задан флаг -i или если ввод и вывод shell связаны с терминалом, то этот shell является интерактивным. В этом случае, сигнал завершения TERMINATE игнорируется ( так что команда kill 0 не вызовет прекращение работы интерактивного shell) и сигнал прерывания INTERRUPT перехватывается и игнорируется( так что команда wait прерываема). Во всех случаях, сигнал выхода QUIT игнорируется shell.

**-r** если задан флаг -r, то работает ограниченный shell. (смотрите описание команды rsh(1)).

Оставшиеся флаги и аргументы приведены в описании команды set приведенной выше.

## СОСТОЯНИЕ ВЫХОДА

При обнаружении ошибок интерпретатором shell, таких как синтаксические ошибки, он возвращает ненулевое состояние выхода. Если shell используется в неинтерактивном режиме, выполнение файла shell прекращается. Иначе, shell возвращает состояние выхода последней выполненной команды. Смотрите описание команды exit приведенное выше.

## ФАЙЛЫ

| | |
|---|---|
| /etc/profile | системный файл по умолчанию profile,если ника- кие другие не присутствуют |
| $HOME/.profile | читается логическим интерпретатором shell при подключении |
| /tmp/sh* | временный файл для символов >> |
| /dev/null | источник пустого файла |

## СМ. ТАКЖЕ

cd(1), env(1), login(5), newgrp (1), rsh(1), test(1), umask(1), dup(2), exec(2), fork(2), pipe(2), signal(S), umask(2), wait(2), a.out(3), profile(5), environ(5).

## ЗАМЕЧАНИЯ

Команда readonly (без параметров) осуществляет такой же вывод как и команда export.

Если используются символы << для обеспечения стандартного ввода в асинхронном процессе, вызванном с помощью символа &, то shell имеет путаную информацию о имени вводного документа; создается файл содержащий "мусор" /tmp/sh* и shell сообщает о невозможности найти этот файл с помощью другого имени.

Если команда выполнена и команда с тем же самым именем установлена в каталоге, входящего в путь поиска, до каталога, где была найдена исходная команда, shell будет продолжать выполнять (exec) исходную команду. Для изменения этой ситуации используйте команду hash.

Если вы покинули текущий каталог или каталог, расположенный выше текущего, то команда pwd не сможет выдать корректный ответ. Используйте команду cd с полным именем пути для изменения этой ситуации.

Когда пользователь команды sh(1) входит в систему, то она читает и выполняет команды из файла /etc/profile до выполнения команд из файла пользователя $HOME/.profile. Поэтому, вы можете изменить среду для всех пользователей команды sh(1) в системе, отредактировав файл /etc/profile.