

Лекция 4. Командная оболочка bash и утилиты командной строки POSIX

Операционные системы

Определения

Определение

Командная оболочка: (интерпретатор командной строки, command line shell, command-line interpreter) — программа, считывающая строки из терминала, текстового файла и других источников и интерпретирующая их в качестве команд некоторой системы.

Режимы исполнения команд

- Интерактивный;
- Пакетный.

Определение

bash: (Bourne-Again shell) — разработана в рамках проекта GNU, впервые выпущена в 1989 г. По умолчанию в GNU/Linux, OS X.

Команды в bash

Синтаксис команды

`<имя> [<ключи>] [<аргументы>]`

Виды команд

- Встроенные;
- Запускающие внешние программы.

Пример

```
$ pwd  
/home/student  
$ /bin/pwd  
/home/student
```

Дополнения команд

Связывание нескольких команд

```
⟨команда⟩ &  
fg  
wait  
⟨команда⟩; ⟨команда⟩
```

Связывание (окончание)

```
⟨команда⟩ && ⟨команда⟩  
⟨команда⟩ || ⟨команда⟩  
⟨команда⟩ | ⟨команда⟩  
(⟨команды⟩)
```

Пример

```
gcc prog.c || { echo "Compilation Error"; exit 1; }  
ps -el | wc -l
```

Дополнения команд (окончание)

Перенаправления

$\langle \text{команда} \rangle [\langle N^o \rangle] < \langle \text{файл} \rangle$
 $\langle \text{команда} \rangle [\langle N^o \rangle] > \langle \text{файл} \rangle$
 $\langle \text{команда} \rangle [\langle N^o \rangle] > | \langle \text{файл} \rangle$
 $\langle \text{команда} \rangle [\langle N^o \rangle] >> \langle \text{файл} \rangle$

Пример

```
mail stu003 < letter.txt  
find /etc -name *.conf > /dev/null  
find /etc -name *.conf 1> etc1.txt 2> etc2.txt
```

Переменные

Переменные

```
⟨переменная⟩=⟨значение⟩  
$⟨переменная⟩  
${⟨переменная⟩}
```

Команда export

```
export ⟨переменная⟩ [=⟨значение⟩]  
export -p
```

HOME IFS PATH

Таблица 1: переменные Bourne Shell

LANG	LC_ALL	LC_COLLATE	...
PWD	RANDOM		

Таблица 2: переменные bash

Примеры использования переменных

Пример

```
$ export LANG=en_US  
$ export LC_ALL=en_US  
...  
$ export LANG=ru_RU.UTF8  
$ export LC_ALL=ru_RU.UTF8
```

Пример

```
$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
$ PATH=${PATH}:/usr/local/mpi/bin  
$ export PATH
```

Команда wc

Синтаксис команды

```
wc [-⟨ключи⟩] [⟨файл⟩ ...]
```

Ключ	Значение
-l	количество строк
-c	количество байт
-m	количество символов
-w	количество слов
-L	длина самого длинного слова

Таблица 3: основные ключи команды wc

Команда ps

Синтаксис команды

ps [-⟨ключи⟩]

Ключ	Значение
-A (или -e)	вывод информации обо всех процессах системы (иначе только процессов текущего пользователя и на текущем терминале)
-l	вывод дополнительной информации о процессах
-F	вывод расширенной информации о процессах

Таблица 4: основные ключи команды ps

Команда kill

Синтаксис команды

```
kill [-⟨номер_или_имя_сингала⟩] ⟨идентификатор_процесса⟩
```

Пример

```
$ ps
  PID TTY          TIME CMD
 3742 pts/0        00:00:01 bash
 3849 pts/0        00:00:00 ps
$ kill -9 3742
```

Команда man

Синтаксис команды

```
man [<ключи>] [<секция>] <страница>
```

Пример

```
$ man kill  
$ man man  
$ man 1 ps
```

Команда echo

Синтаксис команды

```
echo [⟨ключи⟩] [⟨строка⟩ ...]
```

Ключ	Значение
-n	не печатать перевод строки в конце
-e	включить интерпретацию escape-последовательностей, начинающихся с «\»

Таблица 5: основные ключи команды echo

Команда echo (окончание)

Пример

```
$ echo -n 123 > f.txt  
$ echo 456 >> f.txt  
$ echo -e "\ta\tb\tc\n\td" >> f.txt
```

Команда pwd

Синтаксис команды

```
pwd [<ключи>]
```

Пример

```
$ VAR='pwd'  
$ echo $VAR
```

Команда cd

Синтаксис команды

cd [*<ключи>*] [*<каталог>*]

Каталог	Значение
.	текущий каталог
..	родительский каталог
~	домашний каталог
-	предыдущий каталог

Таблица 6: специальные каталоги команды cd

Команда ls

Синтаксис команды

```
ls [<ключи>] [<имена>]
```

Ключ	Значение
-a	также выводить имена файлов, начинающиеся с «.»
-F	добавлять в начало каталогов «/», именованных каналов — « », исполняемых файлов — «*»
-l	выводить подробную информацию (тип, права доступа, количество ссылок, ...)

Таблица 7: основные ключи команды ls

Команда ls (окончание)

Пример

```
$ ls -al
total 16
drwxr-xr-x  4 stu003 stu003 4096 Oct 25 03:41 .
drwxr-xr-x 35 stu003 stu003 4096 Oct 25 02:32 ..
-rw-r--r--  1 stu003 stu003    0 Oct 25 03:41 data.txt
drwxr-xr-x  2 stu003 stu003 4096 Oct 24 22:54 misc
drwxr-xr-x  6 stu003 stu003 4096 Sep 30 13:52 sandbox
```

Команда chmod

Синтаксис команды

```
chmod [⟨ключи⟩] ⟨права⟩ ⟨имена⟩
```

Синтаксис прав

```
[ugoa...][[+|=]][⟨права⟩...][...]
```

Ключ	Значение
-f	не останавливаться при ошибке
-R	выполнять рекурсивно

Таблица 8: основные ключи команды chmod

Команда chmod (окончание)

Пример

```
$ chmod u+x,g-rx,o=r data.txt
$ ls -l data.txt
-rwx---r-- 1 stu003 users 8 Mar 23 14:26 data.txt
$ chmod 770 misc
$ ls -l misc
-rwxrwx--- 1 stu003 users 2 Mar 23 14:24 misc
```

Команда mkdir

Синтаксис команды

```
mkdir [⟨ключи⟩] ⟨каталог⟩ ...
```

Ключ	Значение
-p	не выдавать ошибку, если каталог существует, создавать при необходимости родительские каталоги
-m <i>⟨права⟩</i>	права создаваемого каталога

Таблица 9: основные ключи команды mkdir

Команды rm, rmdir

Синтаксис команд

```
rm [⟨ключи⟩] ⟨каталог⟩ ...  
rmdir ⟨каталог⟩ ...
```

Ключ	Значение
------	----------

-r	рекурсивно удалять каталоги вместе с содержимым
-f	не выводить запросов

Таблица 10: основные ключи команды rm

Команды mv, cp

Синтаксис команд

```
mv [⟨ключи⟩] ⟨источник⟩ ⟨назначение⟩  
cp [⟨ключи⟩] ⟨источник⟩ ⟨назначение⟩
```

Ключ	Значение
-r	рекурсивно копировать каталоги вместе с содержимым
-p	сохранять атрибуты (права доступа и время) при копировании

Таблица 11: основные ключи команды cp

Команды touch, cat

Синтаксис команд

```
touch [<ключи>] <файл> [<файл> ...]  
cat [<ключи>] [<файл> ...]
```

Пример

```
$ touch file1  
$ cat /dev/null > file2
```

Команды head, tail

Синтаксис команд

```
head [⟨ключи⟩] [⟨файл⟩ ...]  
tail [⟨ключи⟩] [⟨файл⟩ ...]
```

Ключ	Значение
-n ⟨номер⟩	вывести первые (последние) n строк файла

Таблица 12: основные ключи команд head и tail

Команда find

Синтаксис команды

```
find [<ключи>] [<каталог> ...] [<выражение>]
```

Ключ	Значение
-L	следовать за символическими ссылками

Таблица 13: основные ключи команды find

Команда find (продолжение)

Определение условия «Больше/меньше/равно»

$\langle \text{условие} \rangle ::= [+|-] \langle \text{число} \rangle$

Выражение	Значение
-name $\langle \text{файл} \rangle$	имена, соответствующие шаблону
-size $\langle \text{условие} \rangle [c]$	размер, равный, больше или меньше заданного
-atime $\langle \text{условие} \rangle$	дата доступа (в сутках)
-mtime $\langle \text{условие} \rangle$	дата изменения
-ctime $\langle \text{условие} \rangle$	дата изменения статуса (владелец, группа, количество ссылок, режим, ...)
-type $\langle \text{тип} \rangle$	тип файла («f» — файл, «d» — каталог, ...)
-user $\langle \text{имя} \rangle$	пользователь
-group $\langle \text{имя} \rangle$	группа

Таблица 14: основные выражения команды find

Команда find (продолжение)

Выражение	Значение
-perm [-]<права>	права доступа совпадают (установлены при "-")
-links <условие>	количество жёстких ссылок
-newer <файл>	время изменения файла больше, чем у заданного
-a	конъюнкция условий
-o	дизъюнкция условий
!	отрицание условия
(...)	порядок
-exec <команда>	выполнить команду, признак окончания — «;», заменитель файла — «{»
-print	печатать полный путь к файлу

Таблица 15: основные выражения команды find (окончание)

Команда find (окончание)

Пример

```
$ find . -name "my*" -type f
$ find /var/ftp/mp3 -name "*.mp3" -type f \
>   -exec chmod 644 {} \;
$ find . -size +100k -a -size -500k
```

Команда grep

Синтаксис команды

```
grep [<ключи>] <строка> [<файл> ...]
```

Ключ	Значение
-i	без учёта регистра
-n	отображать номера строк
-v	отображать строки, <i>не содержащие</i> шаблона
-w	«слово целиком»
-x	точное совпадение строки

Таблица 16: основные ключи команды grep

Команда `grep` (продолжение)

Строка	Значение
<code>^</code>	Соответствует началу строки.
<code>\$</code>	Соответствует концу строки.
<code>.</code> (точка)	Соответствует одному любому символу.
<code>[⟨символы⟩]</code>	Соответствует одному любому символу из перечисленных в скобках.
<code>[^⟨символы⟩]</code>	Соответствует одному любому символу, которого нет в скобках.
<code>⟨символ⟩-⟨символ⟩</code>	Внутри скобок определяет диапазон символов между находящимися слева и справа от «-»: «[a-d]» эквивалентно «[abcd]». Если символ «-» граничит со скобками, он рассматривается как литерал. Например: «[-+]» соответствует «-» или «+».

Таблица 17: элементы регулярных выражений команды grep

Команда grep (продолжение)

Строка	Значение
\(<образец>\)	Задаёт подвыражение.
<образец>*	Соответствует образцу 0 или больше раз.
<образец>\{<число>\}	Соответствует образцу заданное количество раз.
<образец>\{<число>,\}	Соответствует образцу как минимум заданное количество раз.
<образец>\{<число>,<число>\}	Соответствует образцу количество раз в заданном диапазоне.

Таблица 18: элементы регулярных выражений команды grep (окончание)

Команда grep (окончание)

Пример

```
$ grep apple fruitlist.txt  
$ grep ^a.ple fruitlist.txt  
$ grep '\-(ab\)\{1,3\}-'
```

--abab--



--abab--

---ab---



---ab---


--abab----ababab---



--abab----ababab---

-abababab-



 + 

```
$ find . -name "*.xml" -exec grep "ERROR" '{}' \; -print
```


Специальные символы оболочки

Символ	Примеры значения
пробел(ы)	разделитель командной строки
/	разделитель имён каталогов в полном имени
\	экранирование специального значения символов
&	выполнение команды в фоне
;	разделитель команд
	операция конвейера
!	отрицание кода возврата команды
(,)	ограничители группы команд
{, }	ограничители блока кода
"	частичное цитирование
' (апостроф)	полное цитирование
` (обратный)	подстановка команды
<, >	перенаправление ввода/вывода
+	операция сложения

Таблица 19: специальные символы оболочки bash

Специальные символы оболочки (окончание)

Символ	Примеры значения
%	операция остатка от деления
-	перенаправление в стандартный поток ввода/вывода
~ (тильда)	домашний каталог пользователя
?	любой символ в шаблоне
*	любые символы в шаблоне
. (точка)	соответствие любому символу в регулярном выражении
^	соответствие началу строки в регулярном выражении
, (запятая)	разделитель арифметических выражений
\$	подстановка значения переменной, параметра
[,]	определение диапазона символов
#	начало комментария
:	«пустая» команда

Таблица 20: специальные символы оболочки bash (окончание)

Интерпретация символа начала комментария

Пример

```
$ echo ab#cd  
ab#cd  
$ echo ab #cd  
ab
```

Специальные конструкции оболочки

Конструкция	Значение
\	отменяет действие следующего спецсимвола, если в конце строки, то не рассматривается вместе со следующим переводом строки.
'...'	отменяют внутри себя все значения спецсимволов.
"..."	отменяют внутри себя все значения спецсимволов кроме «\$», «'» и «\». «\» сохраняет спецзначение только перед «\$», «'», «"», «\» или концом строки.
'cmd' или \$(cmd)	заменяют стандартный вывод команды.

Таблица 21: основные специальные символы оболочки bash

Специальные конструкции оболочки (окончание)

Пример

```
$ echo "$PWD"
/home/stu003/work/shell
$ echo "`date`"
Fri Apr  6 18:26:58 MSD 2012
$ echo "I'd say: \"Go for it\""
I'd say: "Go for it"
$ echo "\"
> "
"
$ echo "\\\"
\
```

Специальные переменные оболочки

Переменная	Значение
\$?	код возврата последнего конвейера переднего плана
\$_	id последнего исполняемого процесса в фоне
\$*	все параметры сценария (с первого), в виде одной строки
\$@	все параметры сценария (с первого), в виде набора строк
\$#	количество параметров
\$0	имя сценария, как запущен
\$1, \$2, ...	позиционные параметры

Таблица 22: основные специальные переменные оболочки bash

Расширения оболочки (shell expansions)

Пример (фигурных скобок)

```
$ echo sp{el,il,al}l  
spell spill spall
```

Пример (тильды)

```
$ echo ~  
/home/stu003  
$ echo ~root  
/root  
$ echo ~/dir1/  
/home/stu003/dir1/
```

Пример (тильды, окончание)

```
$ cd work  
$ cd dir1  
$ cd dir2  
$ cd dir3  
$ echo ~+  
/home/stu003/work/dir1/dir2/dir3  
$ echo ~-  
/home/stu003/work/dir1/dir2
```

Расширения оболочки (продолжение)

Пример (переменной или параметра)

```
$ VAR1=1
$ VAR11=2
$ echo $VAR11
2
$ echo ${VAR1}1
11
$ VAR1=VAR11
$ echo ${VAR1}
VAR11
$ echo ${!VAR1}
2
```

Пример (команды)

```
$ echo $(date)
Thu Apr 19 19:01:59 MSD 2012
$ echo `date`
Thu Apr 19 19:02:02 MSD 2012
$ DATE=`date`
$ echo $DATE
Thu Apr 19 19:02:12 MSD 2012
```


Расширения оболочки (продолжение)

Пример (арифметического выражения)

```
$ echo $VAR1
VAR11
$ echo $VAR11
2
$ echo $(( VAR1 + VAR11 ))
4
$ echo $[ 3 + 2 ]
5
```

Пример (окончание)

```
$ VAR1=010
$ echo $VAR1
010
$ echo $(( VAR1 ))
8
$ VAR2=0xF
$ echo $(( VAR2 ))
15
$ VAR3=2#101
$ echo $(( VAR3 ))
5
```

Расширения оболочки (продолжение)

Расширение процесса

`<(команда)>, >(команда)>`

Пример (процесса)

```
$ echo <(ls)
/dev/fd/63
$ diff <(ls -l) <(ls -al)
...
```

Расширения оболочки (окончание)

Расширение имён файлов

1 После подстановки:

- параметров;
- команд;
- арифметических выражений

происходит разбиение на слова между пробельными символами (если **не внутри** «"»).

2 Затем в каждом слове ищется *, ?, [. Есть, значит слово рассматривается как образец и заменяется на отсортированный в алфавитном порядке список файлов, удовлетворяющих образцу.

Пример (имён файлов)

```
$ echo .*  
. .
```

Пример (имён файлов)

```
$ rm *.txt
```

Команда if

Команды алгоритмических конструкций

Возвращают код возврата последней выполненной команды или 0, если команды не были выполнены.

Упрощённый синтаксис команды

```
if <команды проверки>; then <команды>; fi
```

Пример

```
$ if ! grep ^$USER: /etc/passwd ; then echo "Non-local user"; fi
dubrov:x:1014:1015:Denis Dubrov,,,:/home/dubrov:/bin/bash
```

Команда “[”

Выражение	Значение
[-a <файл>]	<файл> существует
[-e <файл>]	
[-f <файл>]	<файл> существует и является обычным файлом
[-d <файл>]	<файл> существует и является каталогом
[-h <файл>]	<файл> существует и является символьной ссылкой
[-L <файл>]	
[-p <файл>]	<файл> существует и является именованным каналом
[-b <файл>]	<файл> существует и является блочным устройством
[-c <файл>]	<файл> существует и является символьным устройством
[-S <файл>]	<файл> существует и является сокетом

Таблица 23: параметры команды “[” (проверка типа файла)

Команда “[” (продолжение)

Выражение	Значение
[-r <файл>]	<файл> существует и доступен для чтения
[-w <файл>]	<файл> существует и доступен для записи
[-x <файл>]	<файл> существует и доступен для исполнения
[-s <файл>]	<файл> существует и его размер больше 0
[-N <файл>]	<файл> существует и изменился с момента его последнего чтения

Таблица 24: параметры команды “[” (права доступа и другие свойства файлов)

Команда “[” (продолжение)

Выражение	Значение
[<файл ₁ > -nt <файл ₂ >]	время последнего изменения у <файл ₁ > больше, чем у <файл ₂ > или <файл ₁ > существует, а <файл ₂ > не существует
[<файл ₁ > -ot <файл ₂ >]	время последнего изменения у <файл ₁ > меньше, чем у <файл ₂ > или <файл ₁ > не существует, а <файл ₂ > существует
[<файл ₁ > -ef <файл ₂ >]	<файл ₁ > и <файл ₂ > — ссылки на 1 и тот же файл

Таблица 25: параметры команды “[” (и сравнения файлов)

Команда “[” (продолжение)

Выражение	Значение
[-z <i>⟨строка⟩</i>]	<i>⟨строка⟩</i> пуста
[-n <i>⟨строка⟩</i>]	<i>⟨строка⟩</i> не пуста
[<i>⟨строка⟩</i>]	
[<i>⟨строка₁⟩</i> == <i>⟨строка₂⟩</i>]	<i>⟨строка₁⟩</i> равна <i>⟨строка₂⟩</i>
[<i>⟨строка₁⟩</i> != <i>⟨строка₂⟩</i>]	<i>⟨строка₁⟩</i> не равна <i>⟨строка₂⟩</i>
[<i>⟨строка₁⟩</i> < <i>⟨строка₂⟩</i>]	<i>⟨строка₁⟩</i> лексикографически меньше, чем <i>⟨строка₂⟩</i>
[<i>⟨строка₁⟩</i> > <i>⟨строка₂⟩</i>]	<i>⟨строка₁⟩</i> лексикографически больше, чем <i>⟨строка₂⟩</i>

Таблица 26: параметры команды “[” (проверки строк)

Команда “[” (продолжение)

Выражение	Значение
[<число ₁ > -eq <число ₂ >]	<число ₁ > равно <число ₂ >
[<число ₁ > -ne <число ₂ >]	<число ₁ > не равно <число ₂ >
[<число ₁ > -lt <число ₂ >]	<число ₁ > меньше, чем <число ₂ >
[<число ₁ > -le <число ₂ >]	<число ₁ > меньше или равно, чем <число ₂ >
[<число ₁ > -gt <число ₂ >]	<число ₁ > больше, чем <число ₂ >
[<число ₁ > -ge <число ₂ >]	<число ₁ > больше или равно, чем <число ₂ >

Таблица 27: параметры команды “[” (проверки чисел)

Команда “[” (окончание)

Приоритет	Выражение	Значение
1	! $\langle \text{выр.} \rangle$	истина, если $\langle \text{выр.} \rangle$ ложно
2	\($\langle \text{выр.} \rangle$ \)	истина, если $\langle \text{выр.} \rangle$ истинно
3	$\langle \text{выр.}_1 \rangle$ -а $\langle \text{выр.}_2 \rangle$	истина, если оба $\langle \text{выр.}_1 \rangle$ и $\langle \text{выр.}_2 \rangle$ истинны
4	$\langle \text{выр.}_1 \rangle$ -о $\langle \text{выр.}_2 \rangle$	истина, если $\langle \text{выр.}_1 \rangle$ или $\langle \text{выр.}_2 \rangle$ истинно

Таблица 28: параметры команды “[” (комбинации условий)

Использование команды “[”

Пример

```
$ grep ^$USER: /etc/passwd
dubrov:x:1014:1015:Denis Dubrov,,,:/home/dubrov:/bin/bash
$ if [ $? -ne 0 ] ; then echo "Non-local user"; fi
$
```

Встроенная команда "[[

Отличия команды "[[от "["

- После подстановки переменных нет разбиения на слова (следовательно, нет необходимости помещать \$VAR в «"»).
- Шаблоны в строках не расширяются в имена файлов. Вместо этого операции сравнения строк == и != проверяют соответствие строк образцам.

Пример

```
$ VAR="administration one"  
$ if [[ $VAR == admin* ]]; then echo "You're admin"; fi  
You're admin
```

Команда `if` полностью

Полный синтаксис команды

```
if <команды проверки>;  
then <команды>;  
[ elif <команды проверки>;  
  then <команды>;  
  ... ]  
[ else <команды>; ]  
fi
```

Пример

```
$ YEAR='date +%Y'  
$ if [ ${YEAR% 400} -eq 0 \  
-o ${YEAR% 4} -eq 0 \  
-a ${YEAR% 100} -ne 0 ]; \  
  then echo "yes"; else echo "no"; fi  
yes
```

Команда if с подстановкой арифметических выражений

Пример

```
#!/bin/sh
```

```
YEAR='date +%Y'
```

```
if (( ($YEAR % 400) == 0 || \  
    ($YEAR % 4) == 0 && (YEAR % 100) != 0 )); then  
    echo "$YEAR is leap"  
else  
    echo "$YEAR is not leap"  
fi
```

Команда case

Синтаксис команды

```
case <выражение> in [ <шаблон> ) <команды> ... ] esac
```

Пример

```
case $1 in
  1)
    echo "One"
    ;;
  1?|3?)
    echo "starts with 1 or 3"
    ;;
  *)
    echo "Ends with 6"
```

Пример (окончание)

```
;;
*[3-7])
  echo "Ends with 3 to 7"
  ;;
*)
  echo "Other"
  ;;
esac
```

Команда for по списку

Синтаксис команды

```
for <имя> [ in <список> ]; do <команды> ... done
```

```
# in $@
```

Пример

```
for GUEST in "Mr. Brown" "Mr. Smith" "Mr. Jones"; do  
    echo "Hello $GUEST"  
done
```


Команда for по списку (продолжение)

Пример

```
for NAME in *.sh; do  
    echo "$NAME"  
done
```

Команда for по списку (окончание)

Пример

```
for I in `seq 1 10`; do  
    echo -n " $I"  
done
```

Пример

```
for I in {1..10}; do  
    echo -n " $I"  
done
```

Пример

```
for I in $(seq 1 2 20); do  
    echo -n " $I"  
done
```

Пример

```
for I in {1..20..2}; do  
    echo -n " $I"  
done
```

Команда for с арифметическими выражениями

Синтаксис команды

```
for (( <выражение1>; <выражение2>; <выражение3>; )); do  
    <команды>  
    ...  
done
```

Пример

```
for (( I = 0; I < 20; I += 2 )); do  
    echo -n " $I"  
done
```

Команда while

Синтаксис команды

```
while <команды проверки>; do <команды> ... done
```

Пример

```
NUM=y
```

```
while [ "$NUM" != n ]; do  
    echo -n "Continue? (y/n) "  
    read NUM  
    echo "Processing..."  
done
```

Пример

```
NUM=0
```

```
while read STR; do  
    echo $(( ++ NUM )) $STR  
done \  
    < dir.txt
```

Резервное архивирование

Пример

```
BACKUPDIR=$HOME/backup  
WORKDIR=$HOME/work  
DATE='date +%Y_%m_%d.tar'  
ARCHIVE=$BACKUPDIR/$DATE
```

```
mkdir -p $BACKUPDIR
```

```
find "$WORKDIR" -type f -mtime -5 2> /dev/null | \  
  while read -e FILE; do  
    echo "  Adding $FILE"  
    tar -uf "$ARCHIVE" "$FILE"  
  done
```

Резервное архивирование

Пример

```
BACKUPDIR=$HOME/backup  
WORKDIR=$HOME/work  
DATE='date +%Y_%m_%d.tar'  
ARCHIVE=$BACKUPDIR/$DATE
```

```
mkdir -p $BACKUPDIR
```

```
for FILE in $(find "$WORKDIR" -type f -mtime -5 2> /dev/null); do  
    echo "  Adding $FILE"  
    tar -uf "$ARCHIVE" "$FILE"  
done
```

Команды `until`, `break`, `continue`, `exit`

Синтаксис команды `until`

```
until <команды проверки>; do <команды> ... done
```

Синтаксис команды `break`

```
break
```

Синтаксис команды `exit`

```
exit [ <код_возврата> ]
```

Синтаксис команды `continue`

```
continue
```