

# Автоматизация

## оболочка bash

bash — самая популярная командная оболочка Unix (*другие оболочки sh, csh, ksh*)

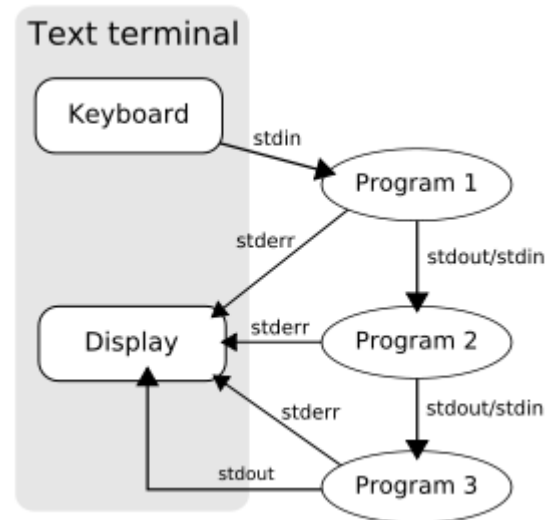
*Расш. Bourne-again-shell (Bourn shell: sh)*

Режимы работы:

- Интерактивный (уже знаком вам)
  - автодополнения по TAB
  - История введенных команд
  - Поиск по истории (Ctrl-R)
  - Поддержка unix pipes
- Обработка сценариев (скриптов), в простейшем случае последовательность команд, записанная в файл +
  - Переменные
  - Условные операторы
  - Циклы
  - Базовые операции со строками и целочисленными типами
  - Массивы
  - Вызов внешних команд и запись результатов их выполнения в переменные
  - Поиск и замена подстрок, регулярные выражения (через sed или expr)
  - Операции над действительными числами (через awk)

# Unix pipes (конвейер)

- Способ перенаправить поток вывода одной команды на вход другой команды
- Стандартные потоки вывода
  - stdout (1)
  - stderr (2)
- Стандартный поток ввода
  - stdin (0)
- Можно создавать программно - системный вызов `pipe()`
- Чтение/запись в/из сокетов (netcat, socat)



**Syntax:** `program1 | program2 | program3`

**Example:** `ls | wc -l`

**Перенаправление вывода в файл:**

`ls > out`

`ls mydir 1>file_out 2>file_err`

`ls mydir &>file_err_out` # перенаправление вывода и ошибок в один и тот же файл

# Bash-script

## Переменные

```
a="Hello world" # присвоение значения переменной  
echo $a # обращение к переменной  
Hello world
```

```
a=$(echo Hello world) # запись текстового вывода команды в переменную  
echo $a  
Hello world
```

## Целочисленная математика (работа с действ. числами чз **awk**)

```
VAR=55 # Устанавливаем переменную VAR, равной 55  
((VAR = VAR + 1)) # Добавляем единицу к переменной VAR.  
((VAR+=1)) # Сокращённая форма записи инкремента  
((++VAR)) # Другой способ. Выполняет префиксный инкремент  
((VAR++)) # Другой способ. Выполняет постфиксный инкремент  
echo $((VAR * 22)) # Умножаем VAR на 22 и передаем результат команде  
echo ${VAR * 22} # Устаревший способ сделать то же  
((VAR<<3)) # Побитовый сдвиг влево (то же, что VAR*8)  
((VAR>>3)) # Побитовый сдвиг вправо (то же, что VAR/8)
```

## Операции со строками

```
a=1234567890  
echo ${#a} # длина строки: 10  
echo ${a:1:3} # 234 (синтаксис ${string:position:length})  
echo ${a:${#a}-3} # 890  
# поиск и замена подстрок, рег. выражения чз. утилиту expr и/или sed
```

# Bash-script

## Условный оператор

```
T1='foo'
T2='bar'
if [[ $T1 == "$T2" ]]
then
    echo 'условие выполняется'
else
    echo 'условие не выполняется'
fi
```

## Циклы

```
COUNTER=0
while [[ $COUNTER -lt 10 ]]
do
    echo The counter is $COUNTER
    let COUNTER=COUNTER+1
done
```

```
for i in "Номер" {1..10}
do
    echo "$i"
done
for i in "Номер" {1..10} ; do echo $i ; done # то же в одну строчку
```

# Bash встроенные переменные

## Аргументы скрипта

<code>\$\$</code>	pid текущего shell (самого процесса-сценария)
<code>\$!</code>	pid последнего процесса в фоновом режиме
<code>\$?</code>	код возврата последнего процесса (функции или скрипта)
<code>\$x</code>	где <i>x</i> — номер параметра, переданного скрипту ( <code>\$1</code> , <code>\$2</code> и т. д., <code>\$0</code> — последний запущенный скрипт)
<code>\$#</code>	количество аргументов командной строки
<code>\$*</code>	все <sup>[7]</sup> аргументы в виде одной строки (слова)
<code>@</code>	то же самое, что и <code>\$*</code> , но при этом каждый <sup>[7]</sup> параметр представлен как отдельная строка (слово)

## Встроенные переменные

(основное)

`$HOME`

`$PWD`

`$HOSTNAME`

`$PATH`

# Bash встроенные переменные

(подробнее)

\$BASH	путь к исполняемому файлу bash
\$BASH_VERSION[n]	массив, состоящий из 6 элементов, содержащий информацию о версии bash
\$BASH_VERSION	версия Bash, установленного в системе
\$DIRSTACK	содержимое вершины стека каталогов
\$EDITOR	заданный по умолчанию редактор
\$EUID	«эффективный» идентификационный номер пользователя (Effective User ID)
\$FUNCNAME	имя текущей функции
\$GLOBIGNORE	перечень шаблонных символов, которые будут проигнорированы при выполнении подстановки имён файлов (globbing)
\$GROUPS	группы, к которым принадлежит текущий пользователь
\$HOME	домашний каталог пользователя
\$HOSTNAME	сетевое имя хоста
\$HOSTTYPE	тип машины (идентифицирует аппаратную архитектуру)
\$IFS	разделитель полей во вводимой строке
\$LC_COLLATE	задаёт порядок сортировки символов, в операциях подстановки имён файлов и в поиске по шаблону
\$LC_CTYPE	определяет кодировку символов
\$LINENO	Номер строки исполняемого сценария
\$MACHTYPE	аппаратная архитектура
\$OLDPWD	прежний рабочий каталог
\$OSTYPE	тип операционной системы
\$PATH	путь поиска (включает в себя каталоги /usr/bin/, /usr/X11R6/bin/, /usr/local/bin и т. д.)
\$PIPESTATUS	Код возврата канала (конвейера)
\$PPID	PID (идентификатор) родительского процесса
\$PS1	приглашение командной строки
\$PS2	вторичное приглашение командной строки, выводится тогда, когда от пользователя ожидается дополнительный ввод. Обычно отображается как «>»
\$PS3	третичное приглашение, выводится, когда пользователь должен сделать выбор в операторе select
\$PS4	приглашение четвёртого уровня, выводится (в изменённом виде) в начале каждой строки отладочного вывода тогда, когда сценарий вызывается с ключом -x . Обычно отображается как «+», «++» и т. д.
\$PWD	рабочий (текущий) каталог
\$REPLY	переменная по умолчанию, куда записывается ввод пользователя, выполненный с помощью команды read
\$SECONDS	время работы сценария (в секундах)
\$SHELLOPTS	список допустимых опций интерпретатора (доступна только для чтения)
\$SHLVL	уровень вложенности shell

# awk – мощный инструмент для разбора строк

- Поточковая обработка
- Входной поток рассматривается как список записей.
- Каждая запись делится на поля.
- На основе этой информации выполняется некоторый определённый программистом алгоритм обработки

Синтаксис:

```
awk [-F 'field_separator'] 'commands' inputfilename
```

или через pipe: `cat inputfilename | awk [-F 'field_separator'] 'commands'`

Пример: `ls -l | awk '$5>10240 {print $9}'` # что делает эта команда?

# awk - встроенные переменные

Переменная	Содержание	По умолчанию
ARGC	Число аргументов командной строки	
ARGV	Массив аргументов командной строки	
ENVIRON	переменные окружения (массив)	
FILENAME	Обрабатываемый входной файл	
FNR	Номер записи в текущем файле	
FS	Разделитель полей записи на вводе	пробел(ы) и/или табуляция
NF	Число полей в текущей записи	
NR	Номер записи (общее число считанных записей)	
OFMT	Формат распечатки чисел	%.6g
OFS	Разделитель полей записи на выводе (символ)	пробел(ы) и/или табуляция



# awk - примеры

## Суммарный размер файлов в папке

```
ls -l | awk 'BEGIN{sum=0}NF>=8{sum+=$5}END{print sum}'
```

## Вычисление среднего и дисперсии для каждого столбца файла

```
{ for(i=1;i<=NF;i++) {total[i]+=$i ; sq[i]+=$i*$i ; } }  
END {  
  
    for(i=1;i<=NF;i++) printf "%f ",total[i]/NR ;  
    printf "\n" ;  
    for(i=1;i<=NF;i++) printf "%f ",sq[i]/NR-(total[i]/NR)**2 ;  
    printf "\n" ;  
}
```

# sed - неинтерактивный потоковый текстовый редактор (stream editor)

## Замена подстроки

```
sed -e 's/oldstuff/newstuff/g' inputFileName > outputFileName
```

```
echo 123123 | sed 's/1/5/g' # output: 523523
```

```
echo 123123 | sed 's/1/5/' # output: 523123
```

## Регулярные выражения

```
sed -e '/^\s*$/d' inputFileName # удаляет пустые строки или строки, с пробелами
```

^ Соответствует началу строки

\$ Соответствует концу строки

. Соответствует любому единственному символу

\* Соответствует нулю или более повторений предшествующего символа

\s Соответствует любому пробельному символу

/xxx/d — удалить строку xxx

# Задачи

- В папке находятся данные в csv файлах. Один из файлов поврежден (число колонок переменное). Найти поврежденный файл.
- Переименовать все файлы в папке, добавив к ним префикс new\_
- Переименовать файлы в папке, содержащие в имени строку 1.0 — надо заменить ее на 2.0
- Найти все фотографии (jpg) в текущей папке и ее подпапках, со временем съемки с августа по октябрь 2018 года по данным EXIF. Создать галерею фотографий, удовлетворяющих указанному критерию, поместив в отдельную папку символьные ссылки на все такие фотографии. Переименовать ссылки таким образом, чтобы они были упорядочены по времени съемки.
- Упорядочить коллекцию mp3 файлов: разложить в папки автор/альбом переименовать файлы в соответствии с шаблоном <номер трека>\_<название композиции>.mp3

Можно использовать:

Awk, bash, exiftool, find, ls, sed, sort, wc, mid3v2, id3tool

# Решение к задаче 1

- В папке находятся данные в csv файлах. Один из файлов поврежден (число колонок переменное). Найти поврежденный файл.

```
cat myfile.csv | awk -F, '{print NF}' # вывод числа колонок в каждой строке  
sort -u # показать уникальные записи  
wc -l # подсчитать число строк
```

#1-й вариант решения

```
for file in *.csv ; do echo -n "$file " ; awk -F, '{print NF}' $file | sort -u | wc -l ; done | awk '$2>1'
```

#2-й вариант решения

```
for file in *.csv ; do awk -F, '{if(NR>1 && lastNF!=NF){print FILENAME; exit}; lastNF=NF}' $file ; done
```