Компьютерная обработка результатов измерений

Практикум 1. Работа с файлами в bash.

Емельянов Эдуард Владимирович

Специальная астрофизическая обсерватория РАН

Лаборатория физики оптических транзиентов

Linux

Ядро. GNU is not UNIX. «Бритва Оккама»: UNIX-way и KISS. «Трубы».
Файловые системы: транзакции, права доступа и аттрибуты файлов, «все есть файл» — псевдофайлы. Монтирование ФС. Команды `mount`, `df`, `du`.
Ссылки на файлы: `ln`.
Структура файловой системы: базовые директории.
Терминалы и псевдотерминалы. Командные оболочки. Команды `man` и `apropos`. Приглашение командной строки: `$PS1`.
Рабочая директория: `pwd` и `$PWD`. Команда `env`. Команда `alias`.

Базовые файловые операции

`ls` – отображение содержимого каталога.
`cd` – переход в другой каталог.
`mkdir` – создать каталог, `rmdir` – удалить.
`rm` – удалить файлы, каталоги.
`find` – поиск файла; `locate` – быстрый поиск с использованием базы данных.
`touch` – создать файл либо изменить атрибуты существующего.
`echo` – вывод в терминал данных.
`reset` – сброс настроек терминала.

Базовые файловые операции

Специальные устройства

`/dev/zero` при чтении возвращает нули.
`/dev/null` уничтожает все данные, которые в него пишут.
`/dev/random` случайные числа (требует «энтропии»).
`/dev/urandom` псевдослучайные числа.

Переменные, скобки, возвращаемое значение

`x="text"; echo "$x"; echo "${x}". echo "'$x'=$x". echo "\$x=$x". unset x. set. env. echo ‘ls‘ и echo $(ls). let a=25+3.`
`${VAR-default}`, `${VAR=default}` – значение по умолчанию.
`${VAR?err message}` – выдача сообщения.
`${VAR:pos[:len]}` – подстрока с `pos` длины `len`.
Математика: `var1=$((5 + 5))`, `var2=$(($var1 * 2))`.
Возвращаемое значение: `$?`. Если возвращается не 0, то это обычно — код ошибки. Например: `команда1 && команда2 || команда3`.
Команды можно объединять: `(команда1; команда2; команда3)` (скобка вернет код возврата последней команды).

Скрипты

Шебанг

`#!/bin/bash` или `#!/bin/sh`. Шебанг необязателен, если скрипт можно вызывать в том же сеансе bash (однако, в случае проблемы сеанс может «упасть»).

Аргументы

`$N` – N-й аргумент (`$0` – имя скрипта). `$#` – количество аргументов.
`$*` и `$@`:

```
function chkargs(){
    echo "you␣give␣$#␣arguments:"
    for arg in "$@"; do
    echo -e "\t$arg"
    done
}
chkargs "$@"
chkargs "$*"
chkargs $*
```

Вывод/вывод файлов, перенаправление вывода

Команда `man`

Выводит справку по флагам различных утилит. `man man`.

Вывод содержимого файла: `cat file`. Перенаправление в другой файл: `cat file1 > file2`.
Номера стандартных дескрипторов: 0 – stdin, 1 – stdout, 2 – stderr:
`cat $file 2>/dev/null`.
`tail -n N` – отображение N строк с конца файла. Чаще с флагом `-f` (для непрерывного перечитывания файла в процессе добавления новых строк).
`head -n N` – отображение N строк с начала файла. Часто — в комбинации с `tail`.
Команды `less` и `more` позволяют интерактивно перемещаться по тексту (доступен также поиск, переход на N-ю строку и т.д.).

Вывод/вывод файлов, перенаправление вывода

Каналы помогают перенаправить вывод одной команды на ввод другой.
Например: `ls -l | less`.
`read` – считать данные со стандартного ввода
`»` позволяет дописывать файл. Например:
`> filelist; while read x; do ls $x » filelist; done`
`exec 1 > myfile` – перенаправить stdout в файл
`exec 2 > errfile` – перенаправить stderr
`exec 2 > &1` – перенаправить stderr в stdout
`exec 0 < file` – читать данные не с stdin, а из файла
Временное: `exec 4 < &0; exec 0 < myfile; ...; exec 0 < &4`

Условия

```
if [ условие ]; then true; else false; fi
[ условие ] && true || false.
```

```
echo "Enter value"
read val
if [ $val -gt 100 ]; then
  echo "value $val greater than 100";
else
  echo "value $val less than 100";
fi
```

```
    echo "Enter filename"
    read f
    [ -d $f ] && echo "$f is a directory"
    [ -f $f ] && echo "$f is a file"
    [ ! -e $f ] && echo "Not exists"
```

case

```
while [ -n "$1" ];do
case "$1" in
-a) echo "Found the -a option" ;;
-b) echo "Found the -b option" ;;
-c) echo "Found the -c option" ;;
*) echo "$1 is not an option" ;;
esac
shift
done
```

Цикл for

```
echo -e "\t1."

for (( a = 1; a < 11; ++a )); do
  echo "a=$a"
done

echo -e "\n\t2."

for a in $(seq 1 10); do
  echo "a=$a"
done

echo -e "\n\t3."

for a in one "two args" three; do
echo "a=$a"
done
```

Цикл while

```
#!/bin/bash

while read X; do
  echo "You entered: $X"
done

echo "End"
```

```
./w
Hello
You entered: Hello
More words
You entered: More words
^D
End
```

Массивы

```
array=(1 2 3 4 [5]=next [10]=last)
echo -n "array with size ${#array[*]} and indexes"
echo " ${!array[*]}: ${array[*]}"
echo "array[4]=${array[4]}, array[10] len=${#array[10]}"
```

Результат:

```
array with size 6 and indexes 0 1 2 3 5 10: 1 2 3 4 next last
array[4]=, array[10] len=4
```

+ скрипт `takeexp`.

Поиск и редактирование в файлах

grep

```
grep take takeexp
echo -e "first line\nsecond line\nafirst line" | grep first
echo -e "first line\nsecond line\nafirst line" | grep -w first
```

Отобразить N линий до, после или вокруг: `-BN`, `-AN`, `-CN`.

Рекурсивный поиск: `-R`.

Инверсия поиска: `-v`.

Вывод номера строки: `-n`, имени файла: `-H`.

Поиск нескольких фраз: `-e фраза`.

Регулярные выражения: `grep [0-9] file`;

IP-адрес: `grep -E "[0-9]{,3}\.[0-9]{,3}\.[0-9]{,3}\.[0-9]{,3}"`.

Конструкция `{min,max}`.

Поиск и редактирование в файлах

sed

```
sed 's/test/another test/g' ./myfile
sed -e 's/This/That/' -e 's/test/another test/' ./myfile
```

Применение результатов к самому файлу: `-i`

Удаление строк: `sed '2,3d' myfile` и по шаблону: `sed '/test/d' file`

Удаление диапазона по шаблону: `sed '/first/,/last/d' file`

Добавить строку до заданной: `sed '5i newline' file`

Добавить после заданной: `sed '5a newline' file`

Заменить строку: `sed '2c newline' file`

Замена отдельных символов: `sed 'y/oldset/newset'`

Вставка файла: `sed '4r file2' file1`

Поиск и редактирование в файлах

awk

Вывод полей с номерами: `awk -F: '{print $1 $4}' file` (-F – разделитель)

```
echo "My name is Tom awk '{$4="Adam"; print $0}'
```

Выполнение команд в начале: `awk 'BEGIN {print "Hello World!"}'`

Команды в конце: `awk 'END {print "End of File"}'`

Использование скриптов в файле: `awk -f awkscript1 /etc/passwd`

Условный оператор: `awk '{if ($1 > 20) print $1}' file`

```
echo -e "10\n20\n30\n40\n50 awk -f awkscript2
```

Математика: `awk 'BEGIN{x=exp(5); print x}'`

Регулярные выражения

Спецсимволы: `.*[]^${}\+?|()` (нуждаются в экранировании).
`^` – начало строки, `$` – конце строки.
Спецклассы: `[[:alpha:]]`, `[[:alnum:]]`, `[[:blank:]]`, `[[:digit:]]`, `[[:upper:]]`, `[[:lower:]]`, `[[:print:]]`, `[[:punct:]]`, `[[:space:]]`.
Символ «или»: `|`. `echo -e "one\ntwo\nthree" | grep -P "one|three"`.
Количество включений: `{{min,max}}`.
Группировка в скобках:
`echo -e "testtest\na test\ntesttesttest" | grep -P "(test){3}"`.
`grep -G` (базовые регулярные) и `grep -P` (расширенные регулярные).
Проверка адреса электронной почты:
`^([a-zA-Z0-9_\-\.\+]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$`

Нестандартные баш-скрипты

```
//usr/bin/gcc $0 && exec ./a.out "$@"

#include <stdio.h>

int main(int argc, char **argv){
  for(int x = 1; x < argc; ++x)
  printf("arg␣%d␣is␣%s\n", x, argv[x]);
  printf("Done\n");
  return 0;
}
```

Запуск:

```
./a.c some "amount␣of" data
arg 1 is some
arg 2 is amount of
arg 3 is data
Done
```

Примеры

1. Получить псевдослучайное число длиной N символов из `/dev/urandom`.
2. Заполнить таблицу в 100 строк с шаблоном: столбец 1 — номер строки, столбец 2 — псевдослучайное число от 0 до 1000, столбец 3 — псевдослучайное число от -20 до 20, столбец 4 — псевдослучайное число с фиксированной точкой от 0 до 100 с 3 знаками после запятой.
3. Отсортировать таблицу из предыдущего примера по 2, 3 и 4 столбцу.
4. В цикле сгенерировать из `/dev/urandom` последовательности латинских букв длиной до 100. Если в последовательности есть искомая (введенная с клавиатуры), отобразить ее на экране. Продолжать до нахождения 5 вхождений или же до достижения 10000 проверок. Вывести на экран количество «попаданий» и «промахов», а также процентную долю «попаданий» по отношению ко всем испытаниям.

Задания

1. Прочитайте `man column`. При помощи этой утилиты отформатируйте в читабельном виде вывод скрипта скрипта из третьего примера. Попробуйте несколько разных видов оформления.
2. Модифицируйте скрипт `takeexp` так, чтобы он брал данные из файла, в котором они хранятся в табличном виде (построчно): номер позиции, время экспозиции в миллисекундах, фокусное расстояние в условных отсчетах.
3. Напишите скрипт, проверяющий, являются ли данные в таблице членами «магического квадрата» 4×4 (сумма по строкам, столбцам и диагоналям должна быть одинаковой).
4. Сгенерируйте 999 случайных целых чисел от 0 до 1000. При помощи `sort`, `head` и `tail` найдите медиану полученного ряда.

Задания

5. Сгенерируйте 10000 случайных целых чисел от 0 до 99. Создайте файл, в который занесите гистограмму распределения чисел (первая колонка — числа, вторая — количество их в ряду данных). Отформатируйте таблицу при помощи утилиты `column`.
6. Нарисуйте в `bash` горизонтальную гистограмму по данным из предыдущего задания (значение фиксированной длины, за которым следует поле из символов * в нужном количестве). Используйте форматированный вывод (например, `printf "%-4d%.5s\n" 123 "**********"`. Попробуйте другой способ нарисовать такую гистограмму. Учтите, что ширина вывода должна должна быть ограничена заданным числом (80 или 100 символов).
7. Используя циклы и массивы, нарисуйте вертикальную гистограмму. Учтите, что рисовать придется сверху-вниз.
8. Почитайте об escape-символах. Попробуйте вывести горизонтальную гистограмму в четырех градациях цвета ($0 \div 25\%$ максимума — одним цветом, $25 \div 50\%$ — другим и т.д.).