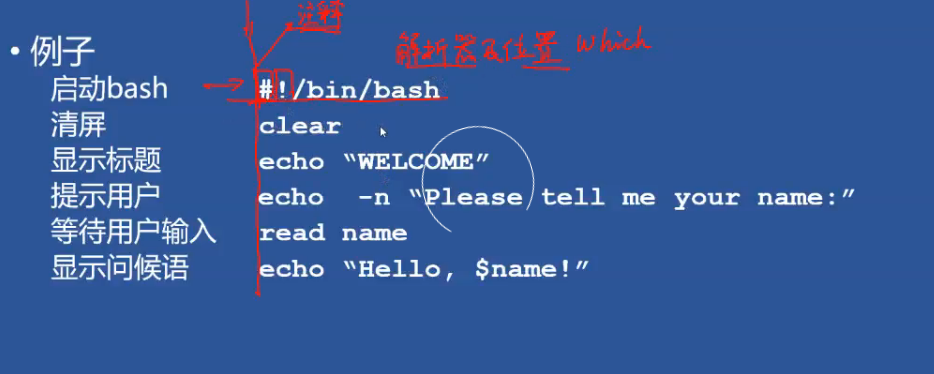
Shell脚本：

10 讲解测验 bash实验

* Shell编程是解释性的 也就是一行执行完在下一行
* #!/bin/bash

1. #表示注释
2. ！表示解析器及其位置，一般存放在 /bin/bash下。找不到就可以用which来找。
3. 写的时候必须顶行，并且是第一行 ,前面不可以加空格

* 举例



1. 下面的语句clear之类的前面可以敲空格啥的
2. Echo 就相当于 printf ；echo –n表示用户后面键入的时候不用换行，跟在后面，注意-n前后都要由空格，注意后面要加空格
3. Read 就是等待用户输入，输入的值赋值给name
4. $取值用的，变量名要取值就需要用$
5. 可执行文件：文件内容需要由CPU可执行的指令来构成，区别于不可执行文件。

脚本不可以叫做可执行文件，他不是指令集，也不是直接由CPU来执行，是由bash解释器来执行。

可执行权限：用户的可执行权限（也就是x）

1. Linux中不声明路径的时候 优先取PATH中找。

* 直接用test0不可以执行脚本，因为找不到，当把文件路径加入到PATH中后(用echo $PATH进行查看)，仍然会报错，因为文件没有可执行权限。所以需要加chmod u+x test0.此时直接运行文件就可以了。

如果把一个脚本文件加入到环境变量PATH中，在给上权限，就可以直接执行。

* 脚本运行方式：

Bash 脚本名字（可以不用后缀） bash hello

Sh 脚本名（带后缀） sh hello.sh

Chmod u+x hello ./hello 给hello执行权限，然后给出hello路径（当前目录下的hello文件，可以直接执行）

这三种都是两个进程 一个bash 一个运行的脚本

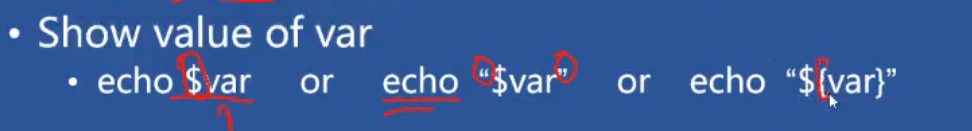
* 重定向： > 可以将结果保存到目标文件中
* 变量：

变量主要考虑三方面：名/类型/值。

Shell里面的数据仅有一种 string类型。，A=10其实本质上A是’10’。计算的时候 expr bc （（））用这些转换数据类型

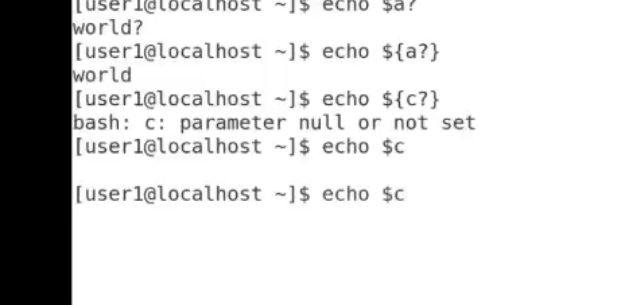
变量无需声明，

* 输出举例



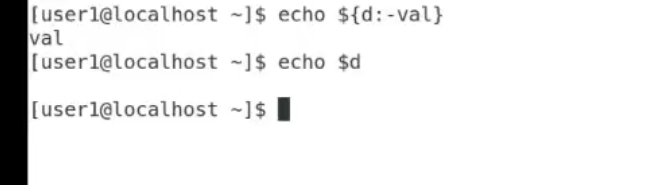
{}用于分隔 如echo ”$as” 输出as变量 echo “${a}s”输出a变量及字符s

1. $a “$a” “${a}”都可以，
2. B=a b=$a 前面是把字符a给b
3. {a?} ？用于显示信息，a有值就无影响返回，a无值就返回报错信息。



1. {a:= }

C有值，就显示c原值，c无值就进行赋值在显示

1. {a:-}

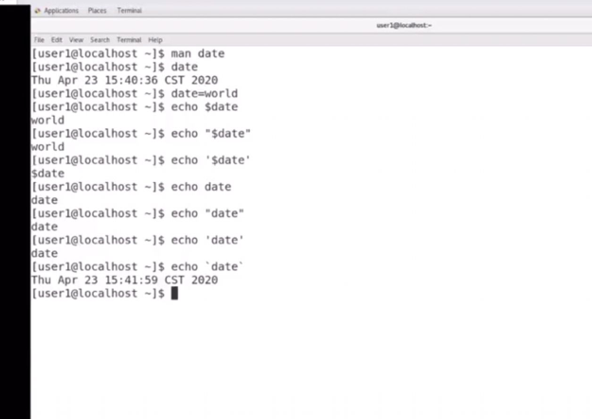
：-不会赋值，只会单次显示

1. {a:?”info”} a不存在返回info
2. {a:+}

有值就会返回value ，无值不操作

* ·反撇号，注意区分单双引号

Echo中输出



单引号括起来都直接输出字符

双引号加$会输出变量值

`某个命令` 会执行该命令 将结果进行输出

* Unset var(在内存中释放掉)

Readonly var(修饰完以后只能进行一次赋值，后面不能再改)

Readonly(直接查看所有的只读变量)

* Echo的一些特殊点

1. -n 不换行用户输入就跟在后面
2. -e 解释转义字符

\b 光标向前移一格在输出

\c -e\c就等于-n

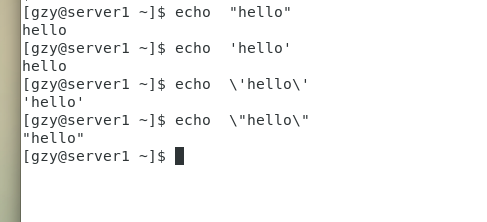
\n 回车并且换行（carriage return回车 line feed换行）

\r 回车但不换行

\v跳一个垂直制表符

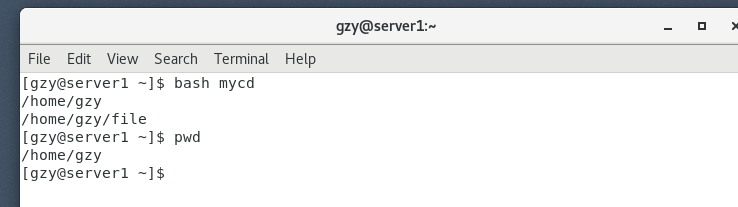
\t 跳一个水平制表符

\” \’ 显示双引号和单引号（还原）



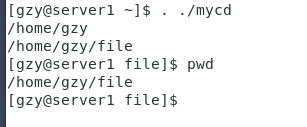
可以把单双引号都还原

* 脚本里面进去了file，而终端这里没有进去。根本原因，脚本和终端（shell）是两个进程，所以脚本执行完毕后，整个进程被杀死，里面pwd变化无影响



* 单进程执行脚本

. ./myscript最开头的是点命令 要空格空开，否则变成上级目录 第二个表示当前目录。



* 环境变量，分为三种：

Env查看所有环境变量

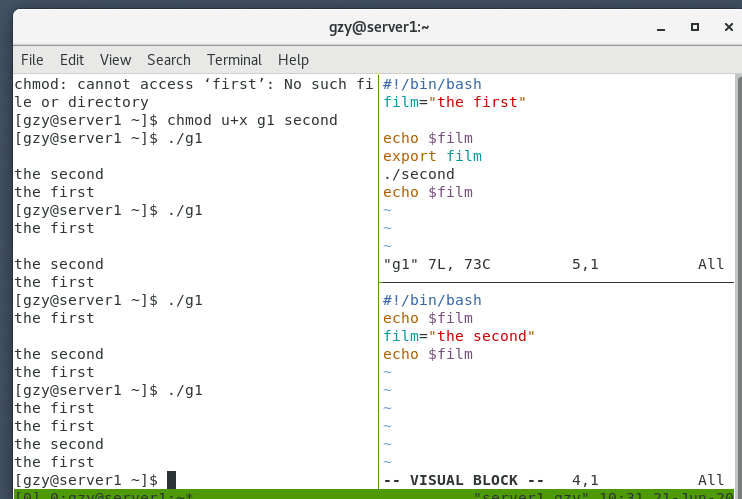
可以用env | more来分页查看

系统级：一般在/ect/profile下 如$HOME $SHELL $PATH

用户级： .bash\_profile（换shell 就会跟着换）

脚本级：只在脚本中产生子进程产生作用

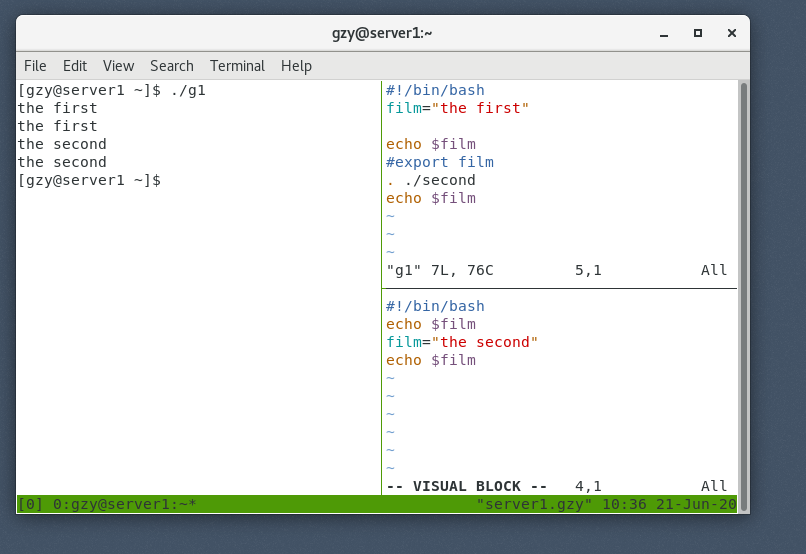
设置环境变量：其余赋值啊之类的类似于普通变量，但是要export command（环境变量要有输出，不能只是对自己产生影响），export是环境变量的核心



**环境变量可以利用export 把父进程中的变量传值给子进程**

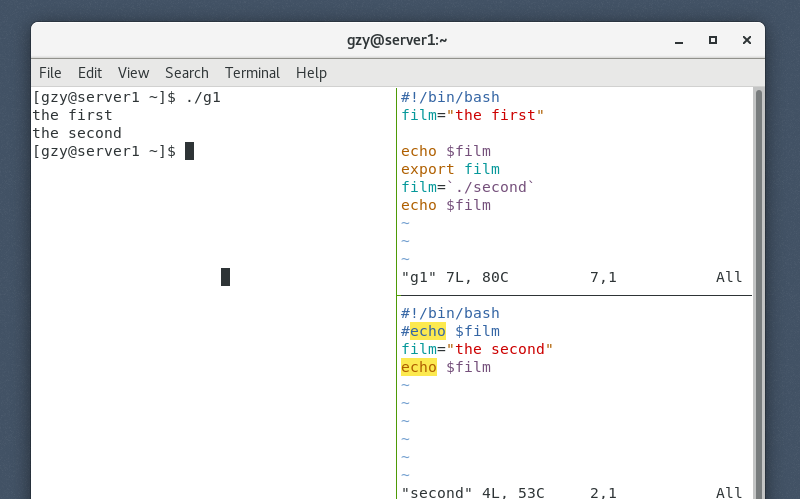
**第一个文件里面 export film进行传值，也就是父进程给子进程，但是子进程传不回给父进程（单向传递的特点）**

**想要子进程传回给父进程，**



**直接点命令**

**或者**



**下面的结果不显示 传给film**

* **位置变量（也就是输入时候的相对位置）**

**传统讲法是九个 $1到$9，但其实现在能有更多还有$0 ${10} ${11} ${…}可以有更多，但是要大括号**

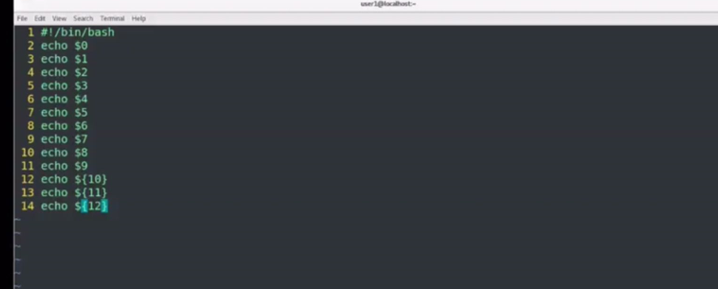
**（此处$和数字整体构成一个变量）**

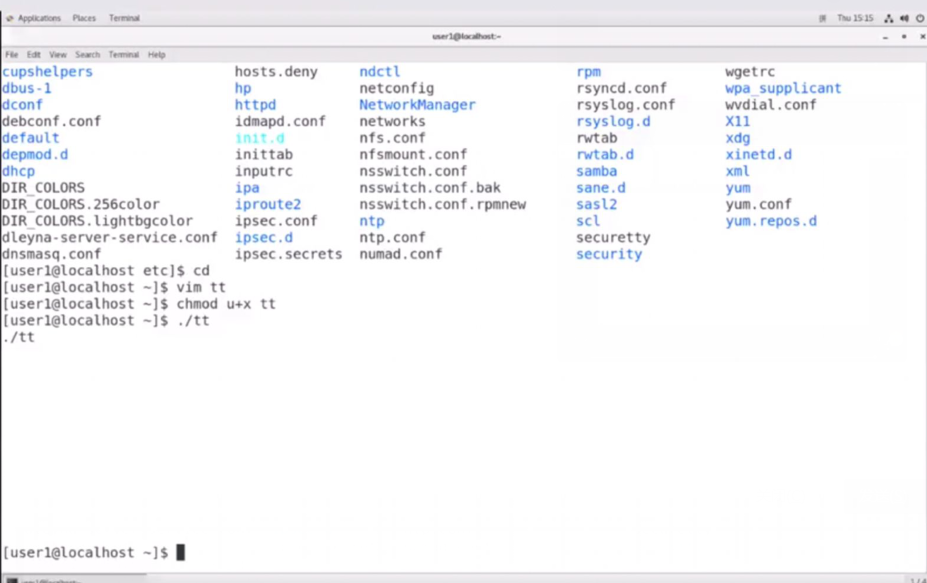
**$0保存命令（也就是你如何执行脚本的命令）**

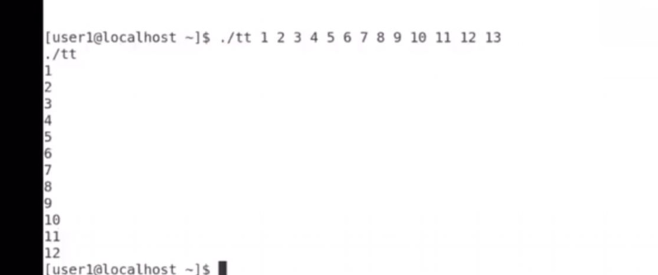
**赋值的时候用**

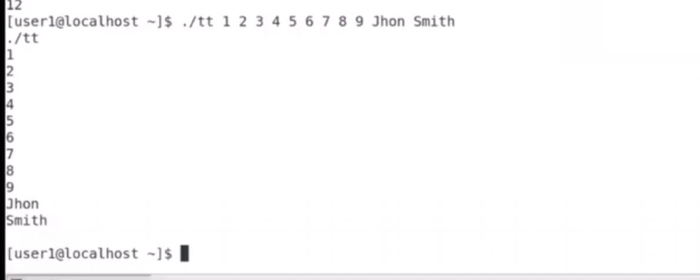
**set + 字符（用空格隔开）**

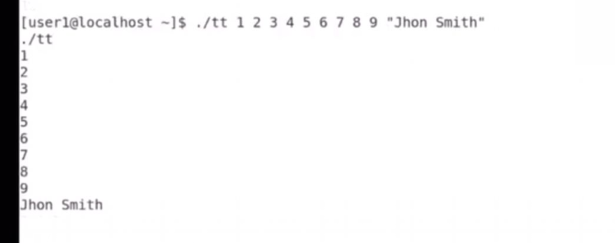
**例如**

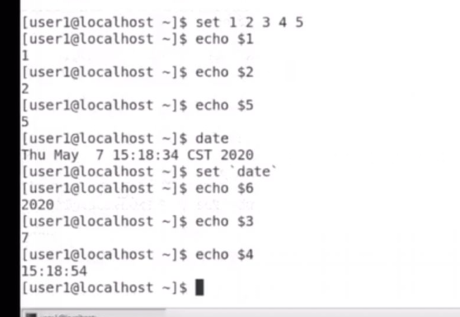


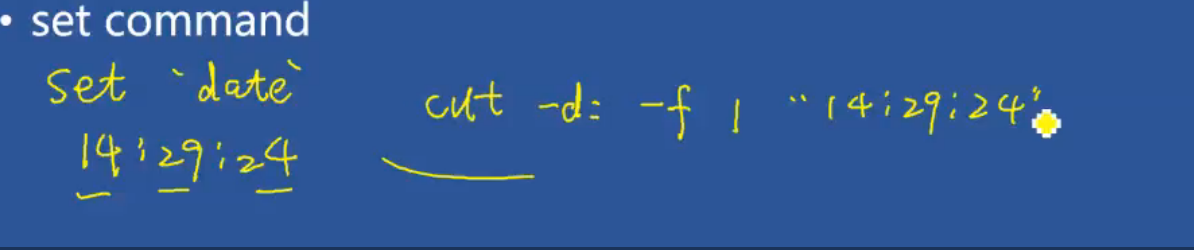








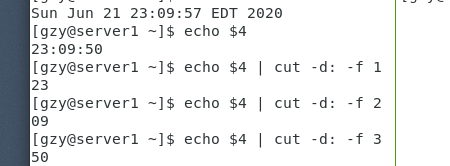




**利用cut -d: -f 1 “14:29:24” -d：表示切割分号 -f 1表示切割完成以后显示哪个域，直接这样写是有问题的，因为cut后面切割的对象需要是一个文件，那么就可以用管道。**

**管道：命令 | …… 左边一定要是一个有输出结果的可执行命令**

**所以此处 echo ”14:29:24” | cut -d: -f 1/2/3**



**想要传文件，就可以用管道送（左边是要用可执行命令的结果）**