bash编程

杨辰@平台技术中心

基本概念

- shell程序: 操作系统和用户之间的桥梁
- 命令输入, 输出, 执行和控制
- 许多内置功能, 可支持编程
- 常见shell软件
  - csh
  - bash (我们使用这个)
  - zsh

基本概念: 输入输出

- 标准流: 0 输入流 1 输出流 2 错误流
- 流重定向
  - 0< file or < file
  - 1> file or > file
  - 2> file
  - 2>&1 错误流重定向到输出流
  - >> 追加模式

基本概念: 管道

- 用法: cmd1 | cmd2
- 原理: 创建临时文件, 将cmd1的输出作为cmd2的输入
- e.g.
  - cat files | sort | uniq | wc -l

基本概念: 任务控制

- 前台/后台任务
  - cmd & 后台执行, 不等待cmd返回
  - 控制命令 jobs / bg / fg / Ctrl-Z
- 进程树模型
  - pstree
- Q: 常驻任务为什么要nohup执行?

BASH编程

- 当作程序语言来看吐槽点太多
- 但是非常实用和强大
- 让我情不自禁的想起了PHP

- BASH编程思想:
  - 分解复杂任务为简单功能
  - 组件间通过文本格式传递消息

- 后面和PHP比较性的介绍

BASH编程: 变量

- 变量类型: 字符串
  - php: 字符串(整数, 浮点数, 字符串), 数组, 对象, ...
- 赋值
  - x=1
  - x = 1 // 错误!
  - php: $x = 1;
- 引用变量 echo $x
- 取消赋值
  - unset x
  - php: unset($x);
- '原子'的处理
  - '原子': echo XXX
  - 解析字符串
  - php: 解析成宏的值 或 字符串(但是报E_NOTICE错误)

BASH编程: 变量&字符串

- " / "" 区别
  - '$x' 不解析变量, "$x" 解析变量
  - php君: 我也是哎
- 字符串连接
  - x=$x / "x=$x" / "xy=${x}y"
  - php君:
    - "x=" . $x
    - "x=$x"
    - "xy=${x}y"
    - "xy={$x}y"
    - ${x} / {$x} 有差异

BASH编程: 环境变量/全局变量

- 一般大写变量名
- $PATH, $PWD, $HOME, ...
- env查看
- 设置环境变量: export XXX=123
- 类比php:
  - $_SERVER, $_GET, $_ENV, ...
  - 唯一全局变量: $GLOBALS
  - 写全局变量: $GLOBALS['XXX'] = 123

BASH编程: 特殊变量

- 命令行相关
  - $# ~ php $argc
  - $@ ~ php $argv
  - $n ~ php $argv[n]
- 命令相关
  - $$ 自身进程号
  - $? 上次执行命令返回码

BASH编程: 整数运算

- 运算环境: ((...)) 或 let "..."
- 得到计算值: $((...))
- 运算符: +, -, *, /, %, **, <<, >>, ++, --, ...
- ++/--运算符可以直接修改变量值
- 未定义变量解析为0
- BASH不支持浮点运算~

BASH编程: 条件判断

- 提示: 命令分隔符为"\n"或者";"
- [...] 或 test ...: 变量比较, 文件类型判断命令
- [[...]] 为bash内置命令, 解析特殊字符, 看上去更像程序:
  - [1 -le 2] VS [[a < b]] ([[支持字符串比较)
  - [true -a false] VS [[true && false]]
  - [! false] VS [[!false]]
  - [[true && (true || false)]]
- man test for more info
- 千万注意格式!!!
- x='1 2'; [$x= "1 2"]; echo $? 错在哪里?

BASH编程: 条件分支

- 命令返回码:
  - 0 => 成功
  - 非0 => 失败
- true: 返回值为0的一条命令
- false: 返回值为1的一条命令
- && / || 条件执行
  - cmd1 && cmd2 && cmd3 仅当前个命令执行成功时才执行下一条命令
  - cmd1 && cmd2 || cmd3 如果cmd1执行成功, 执行cmd2, 否则执行cmd3

BASH编程: 控制流

常见控制流语法:

- if list; then list; [elif list; then list;] ... [else list;] fi
- case word in [[(] pattern [| pattern] ...) list ;;] ... esac
- for name [[in [word ...]] ;] do list ; done
- for ((expr1 ; expr2 ; expr3)) ; do list ; done
- while list; do list; done
- break / continue

- 上述list含义: 命令组合

控制流: if else

```
if true
then
    echo 'Y'
else
    echo 'N'
fi
```

```
if [ -z $x ]; then
    echo 'Y'
else
    echo 'N'
fi
```

推荐格式~

控制流: case

```
case $input in
    a)
        echo a
        ;;
    b)
        echo b
        ;;
    *)
        echo default
        ;;
esac
```

实际执行的是匹配过程

;; 直接退出

;& 继续往下不匹配执行

;;& 继续往下匹配执行

控制流: for in

BASH:

```
for x in $li
do
    echo $x
done
```

PHP:

```
foreach ($arr as $k) {
    echo $k;
}
```

注意列表元素的分隔问题!!!

控制流: for

```
for ((x = 1; x < 10; ++x))
do
        echo $x
done
```

((...; ...; ...)) 特殊解析实现

个人感觉就像php的for, 因为有了foreach, 因此很少用到...

控制流: while

```
while true
do
        sleep
        echo 'hi'
done
```

BASH编程: 命令执行

- 命令名称不含路径前缀时, 查找顺序:
  - alias
  - function
  - 内置命令
  - 根据$PATH变量路径查找
- 命令执行的区别:
  - date +%s 当前shell环境下执行
  - (date +%s) 子shell环境下执行
  - $(date +%s) 将输出替换到当前位置
  - x=$(date +%s) 输出赋值给x
  - "date is $(date +%s)" 也可组合在字符串中
- 从我做起, 请告别`...`的命令执行方式:
  - 不宜读, 尤其和'/"纠缠在一起时
  - 不便嵌套调用: echo $(date +%s -d"$(date)")

BASH编程: alias

- 定义别名: alias timestamp="date +%s"
- 取消别名: unalias timestamp

- 本质是替换字符串执行的过程:
  - alias t1='date +%s'
  - alias t2="t1 -d'2001-01-01'"
  - unalias t1
  - ...

BASH编程: function

```
function timestamp() {
        if [ $# = 1 ]; then
                date +%s
        else
                return -1
        fi
}
```

BASH函数更应该理解为自定义命令:

- 不指定传入参数
- 返回码

BASH编程: function

传入参数的例子:

```
function demo() {
    echo $# $@
    while [[ -n "$1" ]]; do
        echo $1
        shift 1
    done
}
```

BASH编程: 开发和调试

- source / . 导入脚本
  - php: require(...);
- set -x 输出执行流程
- set -u 提示使用未定义变量
  - php: ERROR_NOTICE提示错误
- BASH编程注意:
  - 命令组合伪装成编程语言的形
  - 千万不要被欺骗了
- 千万要有良好的代码习惯

(AddsOn) BASH编程: 字符串处理

- 默认值 ${var:-"default-value"}
- 字符串替换 ${str/search/replace}
- 字符串截断 ${str/0/3}
- 字符串展开 text{,.bak}
- ... (更多奇葩的用法, 等你去发现)