阿里云开发者社区
ALIBABA CLOUD DEVELOPER COMMUNITY

Shell 脚本速查手册

给运维工程师的 Cheatsheets

阿里云开发者学堂推荐配套教材

藏经阁

阿里云开发者电子书系列

阿里云开发者“藏经阁”

海量电子书免费下载

钉钉扫一扫

进入官方答疑群

开 发 者 学 堂 【 Alibaba Cloud Linux 技术图谱】

更多好课免费学

目录

前言

2021 年，阿里云开发者社区联手 Linux 中国开源社区，为广大的运维工程师、开发者提供了一套内容丰富、场景丰富的 Linux 入门课程。本手册为其中的「Shell 脚本入门」、「Shell 脚本进阶」的补充手册，方便学生学习。

这本书适用于所有的 Unix 用户和 Linux 用户。有了这本书，你就可以编写 Bash 脚本，用更短的时间，更轻松、更稳定地完成更多的工作。

本电子书为 Linux 中国开源社区为运维工程师倾心打造，旨在为运维工程师们提供一个快速、便捷的查询手册。本书以本书以普及率最高的 Bash 为基础进行撰写，具体内容组织结构如下：

第一章节介绍 Shell 脚本的编写基础，介绍运维工程师在工作时编写 Shell 脚本的一些基本信息

第二章节介绍 Shell 脚本编写时的一些进阶技巧。

第三章节介绍 Shell 脚本编写过程中一些常用的命令和用法。

第四章节介绍一些常见的 Bash 资源库，帮助运维工程师在编写脚本时，快速实现想要的效果。

Bash 脚本基础

1. Bash 脚本定义变量

在使用 Bash 编写脚本时，你可以根据使用的场景，定义不同类型的变量，从而完成整个脚本的开发。

脚本变量类型

Bash 脚本的变量可以细分为以下三个类型：

(1) 局部变量：局部变量在脚本或命令中定义，仅在当前 shell 实例中有效，其他 shell 启动的程序不能访问局部变量；

(2) 环境变量：所有的程序，包括 shell 启动的程序，都能访问环境变量，有些程序需要环境变量来保证其正常运行。必要的时候 shell 脚本也可以定义环境变量；

(3) shell 变量：shell 变量是由 shell 程序设置的特殊变量。shell 变量中有一部分是环境变量，有一部分是局部变量。

定义变量和使用变量

在定义和使用变量时，应遵循如下描述：

(1) 定义变量：`name=value` 需要注意，等号两侧不能有空格；

(2) 使用变量：`echo $name` 或 `echo ${name}` ；

(3) 定义局部变量：`local name="test"` ；

(4) 定义只读变量：`readonly name`；

(5) 删除变量：`unset name`。

Bash 字符串

在 Bash 中，字符串可以是单引号或双引号，二者的区别如下：

单引号：单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；单引号字串中不能出现单独一个的单引号，但可以成对出现，作为字符串拼接使用；

双引号：双引号里可以有变量且可以出现转义字符。

2. Bash 脚本传递参数

在使用 Bash 脚本时，我们可以传递一些参数给 Bash 脚本，方便 Bash 脚本执行相应操作。具体的参数使用说明如下：

Bash 中的参数

在 Bash 中使用参数时，具体定义如下：

- Bash 中的参数按照数字顺序定义；
- Bash 脚本内获取参数的格式为：**$n**。

几个特殊参数

Bash 中除了正常的顺序参数以外，还有一些特殊参数：

- **$0**：文件名；
- **$#**：传递到脚本的参数的个数；
- **$***：以一个单字符串显示所有向脚本传递的参数。

3. Bash 脚本定义数组

在 Bash 当中，你可以定义脚本来进行批量操作，关于数组的使用，你可以参考下方内容：

定义数组

定义数组：`myArray=(value0 value1 value2)`。

使用数组

使用数组：`${数组名[下标]}`。

获取数组所有元素

`${my_array[*]}` 或者 `${my_array[@]}` 可以获得数组的所有元素。

获取数组长度

`${#array_name[@]}` 或者 `${#array_name[*]}` 可以获得数组长度。

4. Bash 脚本运算符

Bash 为开发者提供了多种运算符，方便开发者进行逻辑运算。

Bash 中的算数运算符

运算符	说明
+	加法
-	减法
*	乘法
/	除法
%	取余
=	赋值
==	相等。用于比较两个数字，相同则返回 true。

运算符	说明
!=	不相等。用于比较两个数字，不相同则返回 true。

关系运算符

运算符	说明
-eq	检测两个数是否相等，相等返回 true。
-ne	检测两个数是否不相等，不相等返回 true。
-gt	检测左边的数是否大于右边的，如果是，则返回 true。
-lt	检测左边的数是否小于右边的，如果是，则返回 true。
-ge	检测左边的数是否大于等于右边的，如果是，则返回 true。
-le	检测左边的数是否小于等于右边的，如果是，则返回 true。

布尔运算符

运算符	说明
!	非运算，表达式为 true 则返回 false，否则返回 true。
-o	或运算，有一个表达式为 true 则返回 true。
-a	与运算，两个表达式都为 true 才返回 true。

逻辑运算符

运算符	说明
&&	逻辑的 AND
||	逻辑的 OR

字符串运算符

运算符	说明
=	检测两个字符串是否相等，相等返回 true。
!=	检测两个字符串是否不相等，不相等返回 true。
-z	检测字符串长度是否为 0，为 0 返回 true。
-n	检测字符串长度是否不为 0，不为 0 返回 true。
$	检测字符串是否为空，不为空返回 true。

文件测试运算符

操作符	说明
-b file	检测文件是否是块设备文件，如果是，则返回 true。
-c file	检测文件是否是字符设备文件，如果是，则返回 true。
-d file	检测文件是否是目录，如果是，则返回 true。
-f file	检测文件是否是普通文件（既不是目录，也不是设备文件），如果是，则返回 true。
-g file	检测文件是否设置了 SGID 位，如果是，则返回 true。
-k file	检测文件是否设置了粘着位(Sticky Bit)，如果是，则返回 true。

操作符	说明
-p file	检测文件是否是有名管道，如果是，则返回 true。
-u file	检测文件是否设置了 SUID 位，如果是，则返回 true。
-r file	检测文件是否可读，如果是，则返回 true。

5. Bash 脚本流程控制

在 Bash 当中，你可以借助流程控制语句完成常见的逻辑控制。

if 控制

if 条件判断可以使用如下格式：

```
if condition
then
    command1
    command2
    ...
    commandN
fi
```

if else 控制

if-else 条件判断可以使用如下格式：

```
if condition
then
    command1
    command2
    ...
```

```
    commandN
else
    command
fi
```

if else-if else 控制

if-else-if else 条件判断可以使用如下格式：

```
if condition1
then
    command1
elif condition2
then
    command2
else
    commandN
fi
```

For 循环

for 循环可以使用如下格式：

```
for var in item1 item2 ... itemN
do
    command1
    command2
    ...
    commandN
done
```

While 循环

while 循环可以使用如下格式：

```
while condition
do
    command
done
```

特殊的 While 循环 - 无限循环

如果你需要实现无限循环，则可以使用如下格式：

```
# 写法 1
while :
do
    command
done

# 写法 2
while true
do
    command
done
```

Until 循环

除了 While 和 For 以外，还可以使用 until 构建循环：

```
until condition
do
    command
done
```

Case (Switch) 控制

如果需要进行分支处理，则可以使用如下格式：

```
case 值 in
value1)
    command1
    command2
    ...
    commandN
    ;;
value2)
    command1
    command2
    ...
    commandN
    ;;
esac
```

跳出循环

在循环过程中，如果需要跳出循环，则可以考虑使用 break 或 continue：

- 跳出循环使用 break；
- 跳过当前循环使用 continue。

Bash 脚本进阶

除了常规的 Bash 使用以外，你还可以使用一些特殊的命令，来完成特定功能。

1. Bash 脚本输出

输出普通字符串

`echo "It is Linux.CN"`。

输出转义字符串

`echo "\"It is Linux.CN\""`。

输出变量

`echo "$name is best Linux Distro"`。

输出换行符

`echo -e "Show me your code! \n"` ，其中 `-e` 用于开启转义。

输出命令执行结果

```
echo `date`
```

格式化输出

使用 `printf` 命令可以进行格式化输出。

printf 支持的格式字符	说明
\a	警告字符，通常为 ASCII 的 BEL 字符

printf 支持的格式字符	说明
\b	后退
\c	抑制（不显示）输出结果中任何结尾的换行字符（只在%b 格式指示符控制下的参数字符串中有效），而且，任何留在参数里的字符、任何接下来的参数以及任何留在格式字符串中的字符，都被忽略
\f	换页（formfeed）
\n	换行
\r	回车（Carriage return）
\t	水平制表符
\v	垂直制表符
\	一个字面上的反斜杠字符
\ddd	表示 1 到 3 位数八进制值的字符。仅在格式字符串中有效
\0ddd	表示 1 到 3 位的八进制值字符

2. Bash 脚本测试

借助测试参数，你可以判断 Bash 中某些语句是否符合特定的条件，比如文件是否存在、字符串长度是否为 0 。测试参数可以帮助你完成更加复杂的 Bash 脚本。

数值测试

参数	说明
-eq	等于则为真

参数	说明
-ne	不等于则为真
-gt	大于则为真
-ge	大于等于则为真
-lt	小于则为真
-le	小于等于则为真

字符测试

参数	说明
=	等于则为真
!=	不相等则为真
-z 字符串	字符串的长度为零则为真
-n 字符串	字符串的长度不为零则为真

文件测试

参数	说明
-e 文件名	如果文件存在则为真
-r 文件名	如果文件存在且可读则为真
-w 文件名	如果文件存在且可写则为真

参数	说明
-x 文件名	如果文件存在且可执行则为真
-s 文件名	如果文件存在且至少有一个字符则为真
-d 文件名	如果文件存在且为目录则为真
-f 文件名	如果文件存在且为普通文件则为真
-c 文件名	如果文件存在且为字符型特殊文件则为真
-b 文件名	如果文件存在且为块特殊文件则为真

3. Bash 脚本函数

函数定义结构

```
[ function ] funname [()]
{
    action;
    [return int;]
}
```

其中

- function 关键词可带可不带；
- funcname 根据实际需要定义；
- return int 根据需要加入。

函数参数

- 调用函数时传递的参数可以在函数中以 $1、$2 的方式获取；
- 使用 $1 的方式获取参数仅可用于前 10 个参数，超出需要使用 $n 获取；

- 特殊参数处理如下：

参数处理	说明
$#	传递到脚本或函数的参数个数
$*	以一个单字符串显示所有向脚本传递的参数
$$	脚本运行的当前进程 ID 号
$!	后台运行的最后一个进程的 ID 号
$@	与$*相同，但是使用时加引号，并在引号中返回每个参数。
$-	显示 Shell 使用的当前选项，与 set 命令功能相同。
$?	显示最后命令的退出状态。0 表示没有错误，其他任何值表明有错误。

4. Bash 脚本输出输入重定向

输出重定向是 Bash 脚本的一个强大的功能，借助重定向，可以对 Bash 命令执行的结果进行操作。

Bash 输入输出重定向

命令	说明
command > file	将输出重定向到 file。
command < file	将输入重定向到 file。
command >> file	将输出以追加的方式重定向到 file。
n > file	将文件描述符为 n 的文件重定向到 file。
n >> file	将文件描述符为 n 的文件以追加的方式重定向到 file。

命令	说明
n >& m	将输出文件 m 和 n 合并。
n <& m	将输入文件 m 和 n 合并。
<< tag	将开始标记 tag 和结束标记 tag 之间的内容作为输入。

/dev/null

如果不希望看到输出，可以将输出重定向到 /dev/null。

5. Bash 文件包含

包含文件

```
# 写法 1
. filename    # 注意点号(.)和文件名中间有一空格
# 写法 2
source filename
```

Bash 编写常用命令

以下是编写 Bash 脚本时常用的命令，你可以根据自己的需要进行选择。

1. pwd

pwd 命令是用来获取当前目录的，可以用于基于当前目录进行文件/文件夹操作。

- `pwd` 输出当前所在目录；
- `pwd -P` 输出当前所在命令，并展示所有的软连接指向。

2. sort

sort 命令是用来对内容进行排序的，可以将文本、数字进行排序。

- `sort path/to/file` 对文件内容进行升序排列；
- `sort --reverse path/to/file` 对文件内容进行降序排列；
- `sort --ignore-case path/to/file` 对文件内容进行忽略大小写的升序排列；
- `sort --numeric-sort path/to/file` 对文件内容进行按数字顺序排列；
- `sort --unique path/to/file` 对文件内容进行唯一排列；

3. echo

echo 命令是用来输出内容的，可以配合格式编码输出特定颜色/风格的文字。

- `echo "message"` 输出信息；
- `echo "my path is $PATH"` 输出包含有环境变量的信息；
- `echo "Hello World" >> {{file.txt}}` 在文件尾部追加内容；
- `echo "Hello World" > {{file.txt}}` 移除当前文件内容，并替换为新的内容。

4. read

read 命令是用来获取用户输入内容，即标准输入设备(键盘)输入内容。

- `read $variable` 读入数据，并设置给变量；
- `read -p "Enter your input here: " $variable` 展示提示，并读入数据，设置给变量；
- `while read line; do echo "$line"; done` 按行读取内容，并执行命令。

5. shift

shift 可以用于将函数的参数移除，其他参数向前移动。

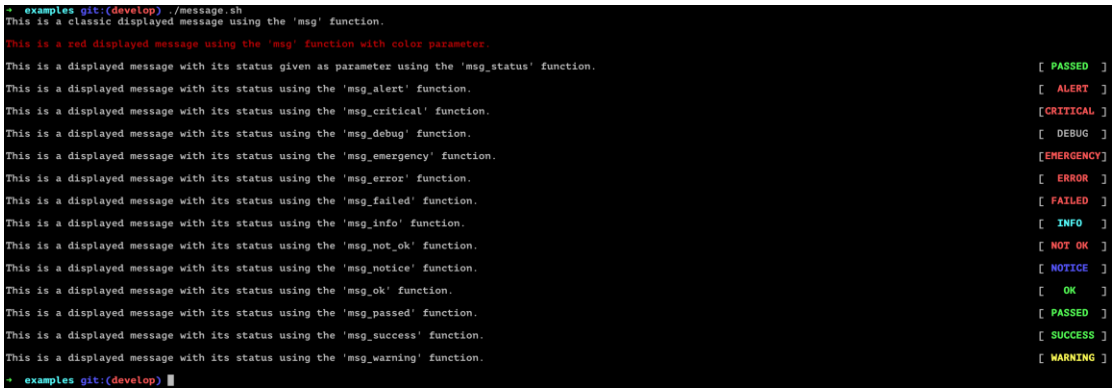
- `shift 3` shift 后可以跟一个数字参数，表示移除相应数量的参数，其他参数向前移动。

Bash 公共库

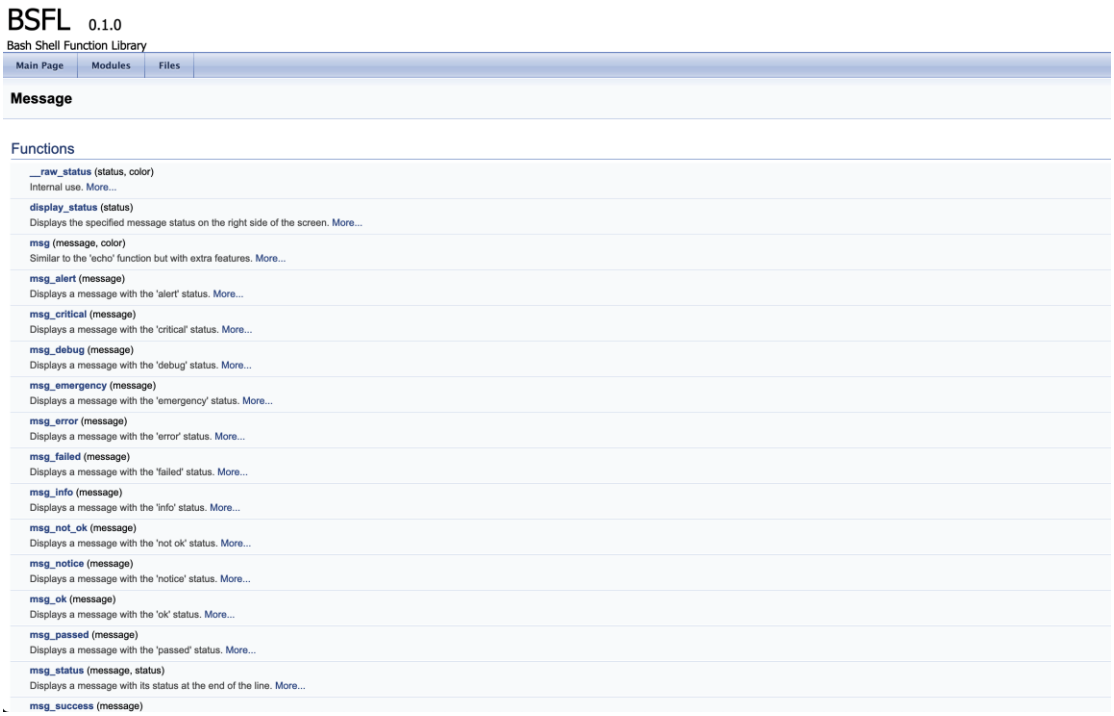
1. Bash Shell Function Library

项目地址：https://github.com/SkypLabs/bsfl

Bash Shell Function Library 是一个短小精炼的 Bash 公共库，他提供了诸如数组操作、命令执行、文件管理、日志记录、信息提醒、网络检测、字符操作、时间操作、变量操作等功能，帮助运维工程师快速完成自己的脚本编写工作。



此外，Bash Shell Function Library 还提供了一个完整的在线文档，帮助你了解函数库中提供的重要函数



使用方法

从 https://github.com/SkypLabs/bsfl 上下载仓库，并获取其中的 bsfl.sh 即可在自己的代码中引用。

示例代码

```
#!/usr/bin/env bash
# -*- tab-width: 4; encoding: utf-8 -*-

declare -r DIR=$(cd "$(dirname "$0")" && pwd)
source $DIR/../lib/bsfl.sh

# --------------------

msg "This is a classic displayed message using the 'msg' function."
echo
```

```
# --------------------

msg "This is a red displayed message using the 'msg' function with color
parameter." "$RED"
echo

# --------------------

msg_status "This is a displayed message with its status given as parameter using
the 'msg_status' function." "PASSED"
echo

# --------------------

msg_alert "This is a displayed message with its status using the 'msg_alert'
function."
echo

# --------------------

msg_critical "This is a displayed message with its status using the 'msg_critical'
function."
echo

# --------------------

msg_debug "This is a displayed message with its status using the 'msg_debug'
function."
echo

# --------------------
```

```
msg_emergency "This is a displayed message with its status using the
'msg_emergency' function."
echo

# --------------------

msg_error "This is a displayed message with its status using the 'msg_error'
function."
echo

# --------------------

msg_failed "This is a displayed message with its status using the 'msg_failed'
function."
echo

# --------------------

msg_info "This is a displayed message with its status using the 'msg_info'
function."
echo

# --------------------

msg_not_ok "This is a displayed message with its status using the 'msg_not_ok'
function."
echo

# --------------------
```

```
msg_notice "This is a displayed message with its status using the 'msg_notice' function."
echo

# --------------------

msg_ok "This is a displayed message with its status using the 'msg_ok' function."
echo

# --------------------

msg_passed "This is a displayed message with its status using the 'msg_passed' function."
echo

# --------------------

msg_success "This is a displayed message with its status using the 'msg_success' function."
echo

# --------------------

msg_warning "This is a displayed message with its status using the 'msg_warning' function."
echo
```

2. Bash Lib

项目地址：http://aks.github.io/bash-lib/

Bash Lib 是一个原子化的公共库，你可以根据自己的实际需要，引入所需的公共库分组，使用相应的内容，降低整个项目的大小。

Bash Lib 提供了诸多原子库，你选择需要使用的引入即可：

- 参数处理：arg-utils
- 日历处理： calendar-utils
- 日期处理：date-utils
- Hash 处理：hash-utils
- 帮助处理：help-util
- 列表处理：list-utils
- 交互处理：prompt-colors
- 文字处理：text-utils
- 测试处理：test-utils
- 时间处理：time-utils
- ...

使用方法

从 https://github.com/aks/bash-lib 下载仓库，并选择你需要引入的脚本，复制到项目目录下，并进行引用即可。

示例代码

```
#!/usr/bin/env bash
source text-utils.sh
lowercase "HELLOWORLD"
```

推荐阅读

- 如何入门 Bash 编程：https://linux.cn/article-13210-1.html
- 编写更好 Bash 脚本的 8 个建议：https://linux.cn/article-6420-1.html
- Linux 中高效编写 Bash 脚本的 10 个技巧：https://linux.cn/article-8618-1.html