# 像黑客一样使用命令行

徐小东

# 像黑客一样使用命令行

徐小东

献给

海燕和铭基

Ħ	录
	-

第一章	人门指引	1
1.1	控制台	1
1.2	终端	3
1.3	终端模拟器	4
	1.3.1 Linux	5
	1.3.2 $\operatorname{macOS}$	6
	1.3.3 Windows	6
1.4	Shell	6
	1.4.1 sh	7
	1.4.2 $\cosh \ldots \ldots$	8
	1.4.3 ksh	8
	1.4.4 bash	8
	1.4.5 $zsh$	9
1.5	命令行界面	9
	1.5.1 功能强大 1	10
	1.5.2 灵活高效	l1
	1.5.3 能自动化 1	L1
1.6	如何进入命令行	l2
	1.6.1 通过控制台进入命令行	l2
	1.6.2 通过终端模拟器进入命令行 1	l2
1.7	你好,命令行	13
kila No		
第二章	神奇补全 1	.5
2.1	何谓补全 1	15
2.2	补全触发按键 1	17

2.3	文件名、路径名补全 17
2.4	程序名、命令名补全 20
	2.4.1 Zsh 自动建议插件 25
2.5	用户名、主机名及变量名补全 26
2.6	可编程补全 31
	2.6.1 bash 示例
	2.6.2 zsh 示例
第三音	重温历史 37
3.1	设置历史变量
3.2	查看历史命令
3.3	搜索历史命令 41
3.4	前后移动历史命令 41
3.5	快速修改并执行上一条命令 42
	3.5.1 删掉多余内容 42
	3.5.2 替换内容 42
	3.5.3 全局替换 43
3.6	快速执行历史命令 44
	3.6.1 重复执行上一条命令 44
	3.6.2 执行以某些字符打头的命令 44
	3.6.3 执行历史列表中第 n 个命令
3.7	快速引用上一条命令的参数 46
	3.7.1 引用最后一位参数 46
	3.7.2 引用最开头的参数 47
	3.7.3 引用所有参数
	3.7.4 引用第 n 个参数 47
	3.7.5 引用从 m 到 n 的参数 48
	3.7.6 引用从 n 到最后的参数 49
3.8	快速引用参数的部分内容 49
	3.8.1 引用路径开头 49
	3.8.2 引用路径结尾 50
	3.8.3 引用文件名 50
	3.8.4 将引用部分更改为大写 51
	3.8.5 将引用部分更改为小写 51
3.9	历史命令展开模式总结

第四章	编辑大法	53
4.1	设置编辑模式	. 53
4.2	Emacs 编辑模式实战	. 54
	4.2.1 按字移动和删除	. 54
	4.2.2 按"词"移动和删除	. 56
	4.2.3 按行移动和删除	. 58
	4.2.4 Emacs 编辑模式总结	. 60
4.3	vi 编辑模式实战	. 60
	4.3.1 移动命令	. 61
	4.3.2 重复命令	. 62
	4.3.3 添加文本	. 62
	4.3.4 删除文本	. 63
	4.3.5 替换文本	. 64
	4.3.6 搜索字符	. 65
	4.3.7 vi 编辑模式总结	. 66
kaka ka	). En lata ste	~
第五章	必备印度	67
5.1	快速导航	. 67
	5.1.1 回到用户王目求	. 67
	5.1.2 回到上次工作的目录	. 68
	5.1.3 访问常用目录	. 69
	5.1.4 自动纠止错误	. 70
	5.1.5 自动导航	. 71
	5.1.6 使用目录栈	. 71
5.2	使用别名	. 73
	5.2.1 定义别名	. 74
	5.2.2 查看别名	. 75
	5.2.3 取消别名	. 75
	5.2.4 别名的缺憾	. 76
5.3	利用 () 构造参数	. 76
	5.3.1 备份文件	. 77
	5.3.2 生成序列	. 77
	5.3.3 连用与嵌套	. 79
5.4	其它妙招	. 80
	5.4.1 命令替换	. 80

	5.4.2	使用变量	1 1 					•				•	•	•	· •		81
	5.4.3	重复执行	行命令			 •	 •	•	 •		•	•	•	•	•	•	82
第六章	周边妇	品															85
6.1	配置框	〔架						•									85
	6.1.1	bash 配	置框架	•				•				•		•			85
	6.1.2	zsh 配置	框架														95
6.2	增强工	具										•					104
	6.2.1	快速路径	2切换:	z.	lua	 •	 •	•	 •		•			•	•	•	105
第七章	结语																111

# 表格

2.1	用户名、主机名及变量名自动补全前缀字符	30
3.1	前后移动历史命令	41
4.1	Emacs 模式按字移动和删除的操作方法	56
4.2	Emacs 模式按"词"移动和删除的操作方法	58
4.3	Emacs 模式按行移动和删除的操作方法	59
4.4	vi 模式移动命令	61
4.5	vi 模式添加文本的命令	62
4.6	vi 模式删除文本的命令	63
4.7	vi 模式复制及粘贴命令	64
4.8	vi 模式替换文本的命令	64
4.9	vi 模式搜索字符的命令	65

# 插图

1.1	IBM 1620 的控制台 2	2
1.2	IBM 1620 控制台的操作前面板 2	2
1.3	Linux 系统虚拟控制台	;
1.4	DEC VT100 终端 4	F
1.5	XTerm 终端模拟器	, )
1.6	Shell 与内核	7
1.7	zsh 的右提示符 §	)
1.8	命令行界面 13	;
2.1	bash 自动补全配置结果 17	,
2.2	在 GIMP 中自动补全文件名	)
2.3	命令自动补全备选列表	)
2.4	命令选项自动补全备选列表 23	;
2.5	zsh 中的命令选项自动补全 24	F
2.6	zsh 中的命令自动建议 26	ì
2.7	bash 中的用户名自动补全备选列表	,
2.8	zsh 中的用户名自动补全备选列表	7
2.9	自动补全的主机名来源	)
2.10	bash 中的变量名自动补全备选列表 29	)
2.11	zsh 中的变量名自动补全备选列表	)
2.12	bash 可编程补全示例	2
2.13	zsh 可编程补全示例 34	Ĺ
3.1	逆向搜索历史命令 41	-
3.2	history 5 执行结果 45	ý
3.3	命令及选项参数编号 48	3

3.4	历史命令展开模式	52
4.1	Emacs 编辑模式图解	60
4.2	vi 编辑模式图解	66
6.1	Bash-it 安装过程	87
6.2	在 Bash-it 中查看别名	88
6.3	在 Bash-it 中查看补全	89
6.4	在 Bash-it 中查看插件	89
6.5	Bash-it 提供的 git 别名	91
6.6	Bash-it 提示符主题	94
6.7	Oh My Zsh 安装过程	96
6.8	Oh My Zsh 插件目录	97
6.9	执行 man zsh 的输出结果	99
6.10	执行 sc-status sshd 的输出结果	99
6.11	Oh My Zsh 的 simple 主题样式	00
6.12	未启用 zsh-syntax-highlighting 时1	02
6.13	启用 zsh-syntax-highlighting 后	03
6.14	执行 zsh_stats 的输出结果 1	04
6.15	更新 Oh My Zsh 1	05

х

# 致谢

感谢 Bash 及 Zsh 开源社区,你们永远是最棒的家伙!

# 更新

你可以从 https://selfhostedserver.com/usingcli-book 获取本书的更新版本。另外,本书也包括视频版本,请通过 https://selfhostedserver.com/usingcli 了解详情。

• Version 2019.3.17: 初版

# 作者简介

徐小东,网名 toy, GNU/Linux 爱好者,DevOps 践行者。喜技术,好分享, 通过 https://linuxtoy.org 网站数年间原创及翻译文章达 3000 余篇。另译 有《Perl 程序员应该知道的事》和《沉浸式学 Git》两本开源图书。Twitter: https://twitter.com/linuxtoy,Mail: xuxiaodong@pm.me<sup>1</sup>。

<sup>&</sup>lt;sup>1</sup>mailto:xuxiaodong@pm.me

# 第一章 人门指引

虽然如今计算机图形化界面大行其道,然而在计算机诞生之初却是命令行界面的天下。在图形化界面中,我们惯常使用鼠标来操作图标或窗口,从而完成各种任务。对于命令行界面来说,情况则有很大的不同。要在命令行界面下执行操作,我们需要更多的依赖键盘。那么,什么是命令行界面呢?在回答这个问题之前,不妨让我们先来谈谈控制台、终端、终端模拟器、以及 Shell 这几个基本概念。

## 1.1 控制台

控制台(Console),又称为系统控制台(System console)、计算机控制台 (Computer console)、根控制台(Root console)、以及操作员控制台(Operator's console)。事实上,早先的控制台是一种用来操作计算机的硬件,如图 1.1 所 示。<sup>1</sup>从这幅图片中,我们可以看到 IBM 1620 计算机的控制台由左边的操作前 面板(参考图 1.2)和右边的打字机组成。通过控制台,操作员将文本数据或待 执行的指令录入到计算机,并最终通过计算机读取或执行。

随着计算机的发展,控制台从硬件概念变成了一个软件概念。于是,控制台有 了新的称呼:虚拟控制台。虚拟控制台正好与物理的控制台硬件相对。通过观 察 Linux 系统的启动过程,我们不难发现:在经过计算机硬件自检之后,一旦 由引导载入程序接管,不一会儿便会进入系统控制台。在这个过程中,通常会 显示如图 1.3 所示的 Linux 系统引导信息。<sup>2</sup>

<sup>&</sup>lt;sup>1</sup>https://en.wikipedia.org/wiki/System\_console#/media/File:IBM\_1620\_Model\_1.jpg <sup>2</sup>https://en.wikipedia.org/wiki/Linux\_console#/media/File:Knoppix-3.8-boot.png



图 1.1: IBM 1620 的控制台



图 1.2: IBM 1620 控制台的操作前面板

#### 1.2 终端



图 1.3: Linux 系统虚拟控制台

## 1.2 终端

跟控制台一样,起初的终端(Terminal)也是一种计算机硬件设备。从外形上 看,终端类似于我们今天所看到的显示器和键盘的结合体。通过终端,用户将 指令和数据输入到计算机。同时,终端也将计算机执行的结果展示给用户。图 1.4 中显示的是曾经广为流行的终端 DEC VT100。<sup>3</sup>

或许你会产生疑问,为什么会出现终端这种硬件设备呢?以今天的眼光来看,显 得似乎有些难以理解。诞生之初的计算机造价相当昂贵,可不像现在人人都能 拥有一台那么简单。除了大型商业组织或大学研究机构,很难在别处看到计算 机的身影。为了能够共享计算机资源,终端应运而生。然而,伴随着科技的进 步,终端最终掉进了历史的黑洞。不过,它后来却以新的形式重生,这就是终 端模拟器 (Terminal emulator),或称之为虚拟终端。

<sup>&</sup>lt;sup>3</sup>https://en.wikipedia.org/wiki/Computer\_terminal#/media/File:DEC\_VT100\_terminal.jpg



图 1.4: DEC VT100 终端

## 1.3 终端模拟器

终端模拟器,即用来模拟终端硬件设备的应用程序。在物理终端中存在的某些显示体系结构,比如用来控制色彩的转义序列、光标位置等在终端模拟器中也得到了支持。图 1.5 显示 Linux 中流行的终端程序之一 XTerm。

不管是 Linux 操作系统,还是 macOS 操作系统,乃至 Windows 操作系统,今 天都有许多终端模拟器可以选择。以下罗列的是这三个操作系统中比较流行的 终端模拟器。

[/home]\$	ls					
vidarlo						
[/home]\$	cd					
[/]\$ cd	etc					
[/etc]\$	ls					
0.0.10.1	n-addr.arpa	csh.cshrc	gshadow-	logrotate.d	odbcinst.ini	rnt
adduser.	conf	csh.login	gtk	lynx.cfg	openoffice	rpc
adjtime		csh.logout	host.conf	nagic	opt	screenro
aliases		db.cache	hostname	mailcap	pam.conf	securetty
alternat	ives	debconf.conf	hosts	mailcap.order	pam.d	security
apn		debian_version	hosts.allow	nailnane	passud	services
apt		default	hosts.deny	mail.rc	passwd-	shadow
asterisk		defona	hotplug	nanpath.config	perl	shadow-
at.deny		deluser.conf	hotplug.d	ndadn	PPP	shells
bakipkun	gfu	dhclient.conf	identd.conf	mediaprm	printcap	skel
bash.bas	hrc	dhclient-script	identd.key	mime.types	profile	squid
bash_com	pletion	dictionaries-common	inetd.conf	mkinitrd	protocols	ssh
bash_com	pletion.d	discover.conf	init.d	modprobe.d	python2.3	sudoers
bind		discover.conf-2.6	inittab	modules	raidtab	sysctl.conf
blkid.ta	ь	discover.d	inputro	nodules.conf	rc0.d	syslog.conf
blkid.ta	b.old	dpkg	ipkungfu	wodules.conf.old	rc1.d	terminfo
calendar		emacs	issue	nodutils	rc2.d	timezone
chatscri	pts	emacs21	issue.net	notd	rc3.d	ucf.conf
chkrootk	it.conf	email-addresses	kernel-img.conf	ntab	rc4.d	updatedb.conf
complete	.tcsh	environment	ldap	wtools.conf	rc5.d	vidarlo.net.hosts
console		exin4	ld.so.cache	Muttre	rc6.d	นวิต
console-	tools	fdnount.conf	ld.so.conf	nysql	rc.d	wgetrc
cron.d		fonts	locale.alias	nanorc	rcS.d	#wvdial.conf#
≣ cron.dai	ly	fstab	locale.gen	network	reportbug.conf	wvdial.conf
cron.hou	rly	groff	localtime	networks	resolvconf	wvdial.conf"
cron.mon	thly	group	logcheck	nsswitch.conf	resolv.conf	X11
crontab		group-	login.defs	<b>UDBCDataSources</b>	resolv.conf"	xpilot
Cron,wee	kly N	gshadow	logrotate.conf	odbc.ini	resolv.conf.pppd-backup	

图 1.5: XTerm 终端模拟器

#### 1.3.1 Linux

- XTerm<sup>4</sup>: XTerm 是 X 窗口环境的默认终端。它提供了与 DEC VT102 和 Tektronix 4014 终端兼容的特性。此外,它也支持 ISO/ANSI 彩色模式。
- GNOME Terminal<sup>5</sup>: GNOME Terminal 是 GNOME 桌面环境的默认终端。它提供了与 XTerm 相似的特性。除此之外,它也包括支持多配置、标签页、鼠标事件等其它功能。
- Konsole<sup>6</sup>: Konsole 是 KDE 桌面环境的默认终端。它包括标签页、多配 置、书签支持、搜索等特性。
- rxvt-unicode<sup>7</sup>: rxvt-unicode 原本克隆自 rxvt,但加入了 unicode 支持, 具有很强的定制特性。另外,rxvt-unicode 还包含 Daemon 模式、嵌入了 Perl 编程语言等功能。本书作者使用的就是这款终端模拟器。

<sup>&</sup>lt;sup>4</sup>https://invisible-island.net/xterm/

<sup>&</sup>lt;sup>5</sup>https://gitlab.gnome.org/GNOME/gnome-terminal/

<sup>&</sup>lt;sup>6</sup>https://kde.org/applications/system/konsole/

<sup>&</sup>lt;sup>7</sup>http://software.schmorp.de/pkg/rxvt-unicode.html

#### 1.3.2 macOS

- Terminal.app: Terminal.app 是 macOS 操作系统默认的终端。它的功能 不多,除了提供设置 TERM 环境变量的选项外,还包括能够使用其搜索功 能来查找 Man pages。
- iTerm2<sup>8</sup>: iTerm2 是 macOS 系统上针对默认终端的开源替代品。它非常 流行,包含许多很棒的功能,比如窗口分割、自动补全、无鼠拷贝、粘贴 历史等等。如果你在 macOS 上工作,那么不妨使用 iTerm2 这款终端模 拟器,相信它所具有的功能一定不会让你失望。

#### 1.3.3 Windows

- Mintty<sup>9</sup>: Mintty 是一个支持 Cygwin、MSYS、WSL 等多种环境的终端 模拟器。它的功能与 XTerm 兼容,包括 256 色和真彩色、unicode、以及 Emoji 表情支持。
- ConEmu<sup>10</sup>: ConEmu 是 Windows 上一款相当流行的开源终端模拟器。它 包含标签页、多种图形窗口模式、用户友好的文本块选择等功能。

## 1.4 Shell

Shell 是一种命令解释程序,它负责用户输入命令的读取、解析和执行。现代 Shell 除了具有与用户直接交互的特性之外,通常也包含编程功能,支持变量、 数组、函数、循环、条件等编程基本要素。

Shell 之所以如此称呼,是由于它相对 Unix 及 Linux 的核心——内核 (Kernel) 而言,处于整个操作系统的最外层,就像乌龟的壳一样。也正因为如此, Shell 提供用来访问系统服务的用户界面,扮演着与内核交互的角色,如图 1.6 所示。

在 Unix 及 Linux 的发展过程中,出现了许多种 Shell,其中比较知名的包括: sh、csh、ksh、bash、zsh 等等。

<sup>&</sup>lt;sup>8</sup>https://www.iterm2.com/

<sup>&</sup>lt;sup>9</sup>https://mintty.github.io/

<sup>&</sup>lt;sup>10</sup>https://conemu.github.io/



图 1.6: Shell 与内核

#### 1.4.1 sh

sh,即 Bourne shell,它是 Unix 第7版的默认 Shell。Bourne shell 由贝尔实 验室的 Stephen Bourne 开发,于 1979 年发布。随着《Unix 编程环境》(Brian Kernighan 与 Rob Pike 著)一书的出版,sh 变得大为流行。

Bourne shell 早已被后来的 Shell 所取代,现代 Linux 系统中的 sh 通常是符号 链接的某个兼容 Shell。例如,本书作者所用的 Debian 9 里的 sh 为 dash。

```
root@toydroid:~# ls -l /bin/sh
lrwxrwxrwx 1 root root 4 Jan 24 2017 /bin/sh -> dash
```

而在作者的另一个系统 Arch Linux 上, sh 则为 bash。

root@codeland:~# ls -l /bin/sh
lrwxrwxrwx 1 root root 4 Feb 7 15:15 /bin/sh -> bash

#### 1.4.2 csh

csh 是 C shell 的简称,它由 Bill Joy 开发,通过 BSD 得到了广泛的分发。在风格上,开发者将 csh 设计得像 C 编程语言一样,因而由此得名。同时,csh 具有很不错的交互使用体验。后来被其它 Shell 所吸收的诸如历史、别名、目录栈、文件名补全、作业控制等特性均出自 csh。

csh 有一个改进版本叫 tcsh, 目前是 FreeBSD 的默认 Shell。

#### 1.4.3 ksh

ksh 指 Korn Shell,其开发者为 David Korn,在 1983 年公布于世。ksh 遵循 POSIX 标准,能够向下兼容 Bourne shell,整合了来自 C shell 的诸多特性。ksh 的一大亮点是引入了 vi 和 Emacs 风格的命令行编辑模式,使用户完全可 以按照自己的按键习惯操作。此外,在 ksh 中还增加了关联数组的特性。

由于 ksh 最初以私有软件的形式进行分发,从而被限制了传播。代之以出现的 替代品包括 pdksh (public domain ksh,公有域的 ksh)、mksh (后成为 Android 的默认 Shell)等。

#### 1.4.4 bash

bash 作为理查德·斯托曼 GNU 工程的一部分出现,从它诞生之初就是为了用 来取代 Bourne shell (参考 1.4.1 节)。Brian Fox 开发了最初的 bash,首个版 本发布于 1989 年。如今, bash 已变得十分流行,它是大多数 Linux 发行版以 及 macOS 的默认 Shell。此外,通过 WSL (Windows Subsystem for Linux), 在 Windows 10 中也可以安装并使用 bash。 bash 的名称来自于 Bourne-again shell,它也遵循 POSIX 标准,其特性吸收 自 sh、csh、ksh 等多种 Shell。

#### 1.4.5 zsh

zsh 是 Z shell 的简称,最初的版本由 Paul Falstad 所开发,发布于 1990 年。 zsh 极大的扩展了 Bourne shell 的功能,并包含来自 tcsh、ksh、bash 等 Shell 的特性。

在交互用户体验上, zsh 尤其出彩。比如, 它支持对命令的选项进行补全、可 以设置右提示符等, 如图 1.7 所示。

	~/src/usingcli ± master • i
[ 416] _book/ [ 96] css/ [ 416] images/ [ 192] latex/ [ 7.5K] 01-getting-started.Rmd [ 116] _bookdown.yml [ 687] _output.yml [ 848] index.Rmd [ 225] usingcli.Rproj [ 60] usingcli.code-workspace	
4 directories, 6 files →	~/src/usingcli ± master • i

图 1.7: zsh 的右提示符

本书主要讨论 bash 和 zsh 这两种目前市面上最流行的 Shell。

## 1.5 命令行界面

命令行界面(Command-line interface),经常缩写为 CLI,亦即用户输入命令的地方。一旦用户将命令输入完毕并加以提交后,后续对命令的解析以及执行的任务都由 Shell 来完成。

与 CLI 相对的是 GUI,即 Graphical user interface, 意为图形用户界面,它

采用图形化的方式让用户与计算机进行交互。因其具有容易使用的优点,包括 Linux、macOS、Windows 等在内的现代操作系统无一例外都提供了图形用户 界面。

既然图形用户界面要比命令行界面更加易用,那么是否说明可以完全抛弃命令 行界面呢?答案是并非如此。事实上,有经验的用户尤其擅长使用命令行界面, 其理由至少包括以下几个方面。

#### 1.5.1 功能强大

让我们先来看一个例子:

```
xiaodong@codeland:~$ history |
awk '{CMD[$2]++;count++;}END \
{ for (a in CMD)print CMD[a] " " \
CMD[a]/count*100 "% " a;}' |
grep -v "./" |
column -c3 -s " " -t |
sort -nr |
nl |
head -n10
```

在作者的 macOS 系统上执行这条命令后, 其输出结果如下:

1	1348	14.3771%	cd
2	1034	11.0282%	1
3	838	8.93771%	git
4	569	6.06869%	ssh
5	513	5.47142%	cat
6	405	4.31954%	vim
7	372	3.96758%	brew
8	360	3.83959%	scp
9	265	2.82637%	rm
10	264	2.8157%	grep

#### 1.5 命令行界面

这条命令虽然看起来似乎有些"吓人",因为它由 history、awk、grep、column、 sort、nl、head 等 7 个命令组成,并通过管道符(|) 串接在一起;然而其结果 却颇为有趣。它将作者平时在命令行中执行的所有命令都进行了统计,最终展 示出 10 个最常用的命令,并相应列出每个命令的使用次数和所占百分比。

管道符将前一命令的输出作为后一命令的输入,使这些表面上不相干的命令进 行协同工作,犹如搭积木一般。这是命令行的真正威力所在。

#### 1.5.2 灵活高效

再看另一个例子,假如我们打算从 photos 目录中找出今年三月份拍摄的照片, 并将其文件名称保存到 mar\_photos.txt 这个文本文件中。在图形用户界面中, 首先,我们可能会打开一个文件管理器(在 Linux 下也许是 GNOME Files, macOS 中则是 Finder)。接着,导航到 photos 这个目录,同时切换成详细视 图模式。然后,我们睁大双眼逐一找出符合要求的照片。可是,现在怎么把照 片的文件名称写到文本文件中呢?我们当然可以直接输入,或者想省点力使用 复制和粘贴也行。要是找出的文件数量比较多,那可绝对是体力活。

但是,如果在命令行下,那么我们只需通过执行一条命令即可达到目的:

xiaodong@codeland:~\$ cd photos; \
ls -l | grep 'Mar' | awk '{ print \$9 }' > mar\_photos.txt

#### 1.5.3 能自动化

使用命令行还有一个很棒的优势,那就是能够自动化各种操作。Shell 允许我们 将所用的命令编写成函数(Function)或脚本(Script)。这样,我们不仅可以 反复执行它们,而且函数或脚本比手动输入效率更高。由此,我们得以从重复 的劳动中解放出来,从而能够腾出时间去做其它有意义的事情。

xiaodong@codeland:~\$ ./script.sh

### 1.6 如何进入命令行

通过前面的描述,现在你应当了解:我们想要输入命令的界面是由 Shell 提供的。那么,如何执行 Shell 呢?我们可以通过下面两种方法来进入命令行。

#### 1.6.1 通过控制台进入命令行

为了节省系统资源,Linux 服务器通常没有附带图形用户界面。当它启动完毕时,在控制台按照提示输入用户帐号及密码并登录后,所进入的即是命令行界面。以下为Linux 服务器的登录提示:

#### login:

Password:

作为普通用户来说,一般使用的是具有图形用户界面的 Linux 桌面系统。在它 启动后就直接进入了桌面,那么此时想要进入控制台,可以按照下列步骤执行:

- 1. 按 Ctrl + Alt + F1 组合键,进入编号为1的控制台。
- 2. 按 Ctrl + Alt + F2 组合键,进入编号为 2 的控制台。
- 3. 依次类推,可以分别进入3号、4号、5号、以及6号控制台。在默认情况下,Linux一般提供6个控制台。
- 4. 如果要从控制台返回到桌面,则可以按 Ctrl + Alt + F7 组合键。



要是在控制台的丛林中迷失了方向,我们可以执行 tty 命令来了解当前 在哪个控制台。

#### 1.6.2 通过终端模拟器进入命令行

另外一种进入命令行界面的方法是使用终端模拟器。在不同的操作系统中,可以选择的终端模拟器程序也有所不同(参考 1.3 节)。本书作者在 Linux 下常用 rxvt-unicode, macOS 中则使用 iTerm2。

一般而言,终端模拟器程序会跟系统的登录 Shell (或称默认 Shell) 绑定在一起。有些终端模拟器程序提供了更改 Shell 的特性,从而使用户可以方便的选择自己惯用的 Shell。如果不能从终端程序中直接更改 Shell,那么也可以通过 chsh 命令来改变登录 Shell。假如我们想把默认 Shell 更改成 zsh,则可以执行 以下命令:

xiaodong@codeland:~\$ chsh -s /bin/zsh

怎么判断当前所用的 Shell 是哪一种呢? 只需执行 echo \$SHELL 即可。

## 1.7 你好,命令行

在《C程序设计语言》中,作者 Brian W. Kernighan 和 Dennis M. Ritchie 介绍的第一个程序是在屏幕上输出一行"Hello world"的消息。为了说明命令行的使用,我们也将在屏幕上输出类似的消息——"你好,命令行"。

当我们进入控制台或打开终端模拟器时,通常会看到跟图 1.8 相似的命令行界面。

xiaodong@	codeland	:~\$	echo	-e	"\t你好,	命令行"
1	2	<b></b> 3 4			5	

图 1.8: 命令行界面

从图 1.8 中我们可以看到命令行一般由下面几个部分组成:

- 1. 当前登录的用户名称,在本例中是 xiaodong。
- 2. codeland 是主机名称, 跟 hostname -s 的输出一致。
- 当前工作目录, ~ 代表用户的主目录, 在 Linux 系统下也就是 /home/<</li>
   用户名 >, macOS 中则为 /Users/< 用户名 >。
- 4. \$ 为命令提示符。通常普通用户的命令行提示符与超级用户(root)的不同,以 bash 为例, root 用户的命令行提示符为 #。

5. 待执行的命令,在本例中是 echo -e "\t 你好,命令行",除 echo 命令本身外,还包括该命令的选项(-e)以及参数(\t 你好,命令行)等部分。命令的选项参数一般由引号(")引起,以避免诸如空格之类的特殊字符所导致的岐义。可以使用单引号(')或双引号("),但语意会不同。

除了这 5 个部分之外,在这个命令行中,我们还可以看到 @、:、以及、、(空格)等字符。@ 一般用来分隔用户名和主机名,其形式跟电子邮箱地址一样。: 在这里起到提示说明作用。空格则常常用来分隔命令的选项和参数。因为命令 行提示符可以定制,所以你的命令行界面可能跟我们在这里介绍的不同。

现在,请你跟我们一起,在命令行的提示符(\$或 #)后面输入 echo -e "\t 你好,命令行"。如果在输入过程中有错误,不必慌张,按**退格键**(BackSpace) 或**删除键**(Delete)删除后重新输入即可。当所有字符全部输入完成后,按下 **回车键**(Enter)。

发现了什么?命令行向我们回显了一条"你好,命令行"的消息。而且 echo 命 令参数中的 \t 在输出中产生了一个制表符 (Tab),从而让消息有了缩进效果。

xiaodong@codeland:~\$ echo -e "\t 你好, 命令行" 你好, 命令行

恭喜!你刚刚在命令行成功执行了一条命令,是否感觉并没有想象中那么恐怖 呢?在后面的章节中,我们将教你如何更加高效的使用命令行,从而提升你的 工作效率。