Introduction of Linux

汪杰

PART II

- Shell Script
- Compile & Debug (for C)
- Text Editor (Vim, VSCode)

Shell Script

A shell script is a program designed to be run by the shell. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution and printing text. A script which sets up the environment, runs the program, and does any necessary cleanup, logging, etc. is called a warpper .

Variable

Define, Assignment & Read

```
VariableName=value
read VariableName
```

- no space between VarName and the equality sign
- first letter: a-z A-Z
- no keywords of shell

Use a variable

```
$VariableName
${VariableName}
```

Special Variables

```
$0  # filename of the script
$1 to $9 # Arguments to the script. $1 is the first argument and so on.
$#  # the number of the arguments
$?  # Return error code of the previous command
$@  # all command line arguments
$HOME  # user directory
$$  # Process identification number (PID) for the current script
!! # Entire last command, including arguments.
$_ # Last argument from the last command.
```

`!!` : A common pattern is to execute a command only for it to fail due to missing permissions; you can quickly re-execute the command with sudo by doing sudo !!

`$_` : If you are in an interactive shell, you can also quickly get this value by typing `Esc` followed by `.`

See bash manul here for more details

Examples:

test1.sh

```
#!/bin/bash
read a
read b
c=$[($a+$b)**$a]
echo $c
```

with arguments

```
#!/bin/bash
echo $[($1+$2)**$1]
```

Command Substitution

```
foo=$(command)
```

It runs the command, captures its output, and inserts that into the command line that contains the $(...)

```
echo "We are in $(pwd)"
```

String

single quotes

```
str='no variables'
```

double quotes

```
v='variables'
str="$v or \"escape character\""
```

connecting

```
str1="connecting strings"
str2="simple"
str3=$str1" is "$str2
```

string length

```
${#string}
```

substring

```
${string:begin:end}
```

Example:

```
#!/bin/bash
str="ECNU is beautiful"
echo ${#str}
echo ${str:1:4}
```

printf

differences from `printf()` in C

- no ( )
- using space between two arguments

if the number of arguments is greater than the number of % in format, the format-string will be reused repeatedly

```
printf "%s %s\n" 1 2 3 4
```

output:

```
1 2
3 4
```

Branches

```
if  [condition]; then
    ...
else
    ...
fi
```

or

```
if  [condition1]; then
    ...
elif [condition2]; then
    ...
else
    ...
fi
```

Conditions

Numerical Comparison Operators

Operator	Remark
-eq	==
-ne	!=
-gt	>
-lt	<
-ge	>=
-le	<=

Other Operators

Operator	Remark
=	== for string
!=	!= for string
-z	If the string is empty
-f / -d	is file / is dir.
-r / -w / -x	check permission
-e	if a file/dir. exists

See `man test` for more infomation

Loop

```
for i in 1 2 3 4 5; do # or: for i in {1..5}; do
    echo $i
done
################################################
# C type for loop, won't work in some POSIX shells
for (( i = 1; i <= 5; i++ )); do
    echo $i
done
################################################
while [ condition ]; do
    ...
done
################################################
# do while loops, until loops, etc.
```

```
break
continue
```

Example:

```
#!/bin/bash

echo "Starting program at $(date)" # Date will be substituted

echo "Running program $0 with $# arguments with pid $$"

for file in "$@"; do
    grep foobar "$file" > /dev/null 2> /dev/null
    # When pattern is not found, grep has exit status 1
    # We redirect STDOUT and STDERR to a null register since we do not care about them
    if [ $? -ne 0 ]; then
        echo "File $file does not have any foobar, adding one"
        echo "# foobar" >> "$file"
    fi
done
```

How to execute bash scripts in shell

- `bash script.sh`, let the `bash` program to run the script for you.
- `./sript.sh`, let the shell to figure out how to run the script.
  - This requires execute permission on script.sh, which can be granted by command: `chmod +x script.sh`.
  - The first line of script.sh must be `#!/bin/bash`, which in fact tells the shell how to run the script.
- `source script.sh`, the shell itself will run the script. Will have side effects. NOT RECOMMENDED if you don't know what you are doing.
- `. script.sh`, in bash, this is identical to `source script.sh`. Also NOT RECOMMENDED!

Static code check

```
shellcheck script.sh
```

`shellcheck` is a static analysis tool that shows warnings and suggestions concerning bad code in bash/sh shell scripts.

It can be installed by

```
sudo apt install shellcheck
```

PART II

- Shell Script
- Compile & Debug (for C)
- Text Editor (Vim, VSCode)

Compilation & Execution

`GCC` (GNU C Compiler → GNU Compiler Collection)

```
gcc test.c      # compile the C source file
```

produce an executable file named (by default) `a.out`

```
./a.out         # run the program a.out
```

Useful Options

```
gcc test.c -o test
gcc test.c -g -o test
```

Separate Compilation

complie a program with several separate files

```
gcc -c test1.c
gcc -c test2.c
gcc test1.o test2.o -o test
```

`-c`: compile to produce an object file, which is not executables just machine-level representations of the source code.

Linking with Libraries

Library

`lib+name.a`	(-static)
`lib+name.so`	(default)
`-l+name`	Link with libraries manually
`-L+lib's dir`	Give the directory manually

```
gcc hello.c -shared -o libhello.so
gcc test.c -lhello -L. -o test
export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
```

```
gcc hello.c -c -o hello.o
ar -r libhello.a hello.o
gcc test.c -lhello -L. -static -o test
```

Debugging with GDB (GNU debugger)

Command	Remark
file [file name]	load a executable file
r	run
c	continue
b [line number] b [function name]	set Breakpoint
s, n	execute a line of source code
p [variable name]	print the value of a variable
q	quit
help [command]	

PART III

- Shell Script
- Compile & Debug (for C)
- Text Editor (Vim, VSCode)

Vim

Vim is a highly configurable text editor built to make creating and changing any kind of text very efficient.

Installation for Linux

via Package Manager(apt-get)

```
sudo apt-get install vim
vimtutor # obtain a vim tutorial
```

Creat a file

```
vim filename
```

Three Modes

Command Mode

all the keys are bound to commands (typing "j" -- it will move the cursor down one line)

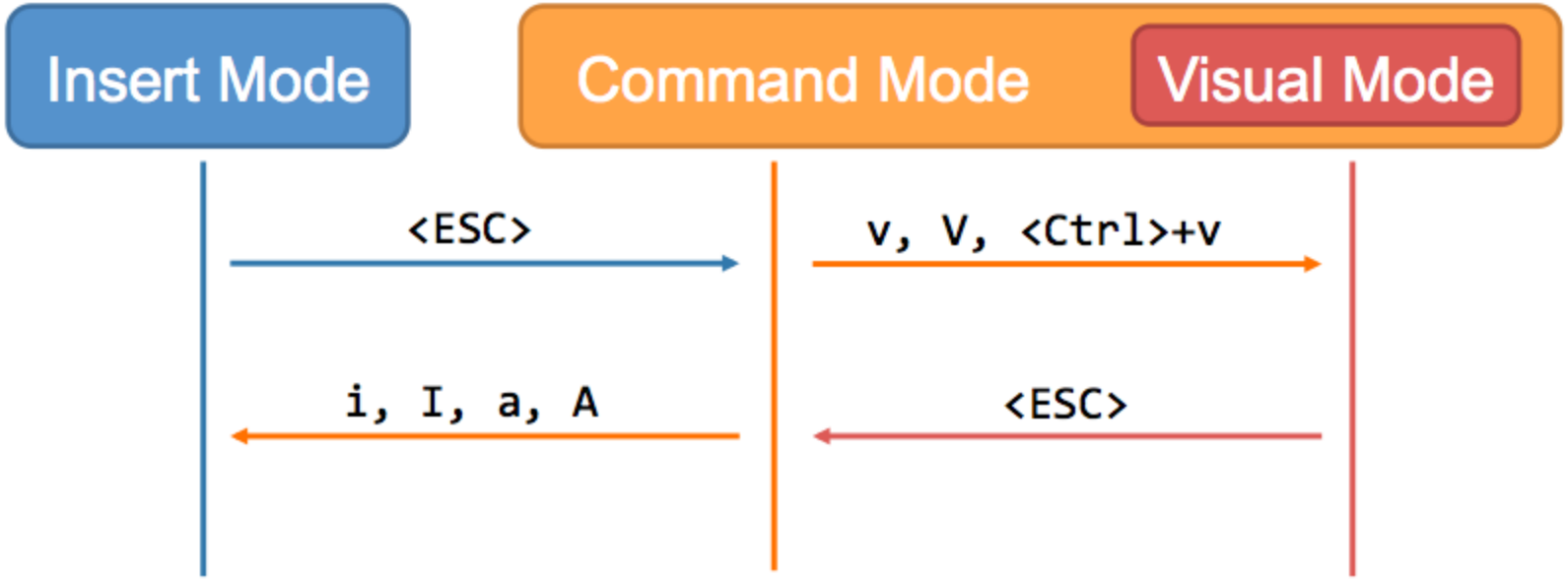
Insert Mode

all the keys are exactly keys (typing "j" -- inserting "j")

Visual Mode

helps to visually select some text, may be seen as a submode of the the command mode

Three Modes



Keys in command mode

Quit and Save

- `:w` write the current buffer to disk (save)
- `:q` quitclose the current window
- `:x` or `:wq` save and close
- `:q!` close without save

Scroll the Screen

downwards

- *ctrl + f* 1 page (forward)
- *ctrl + d* 1/2 page (down)
- *ctrl + e* 1 line

upwards

- *ctrl + b* 1 page (back)
- *ctrl + u* 1/2 page (up)
- *ctrl + y* 1 line

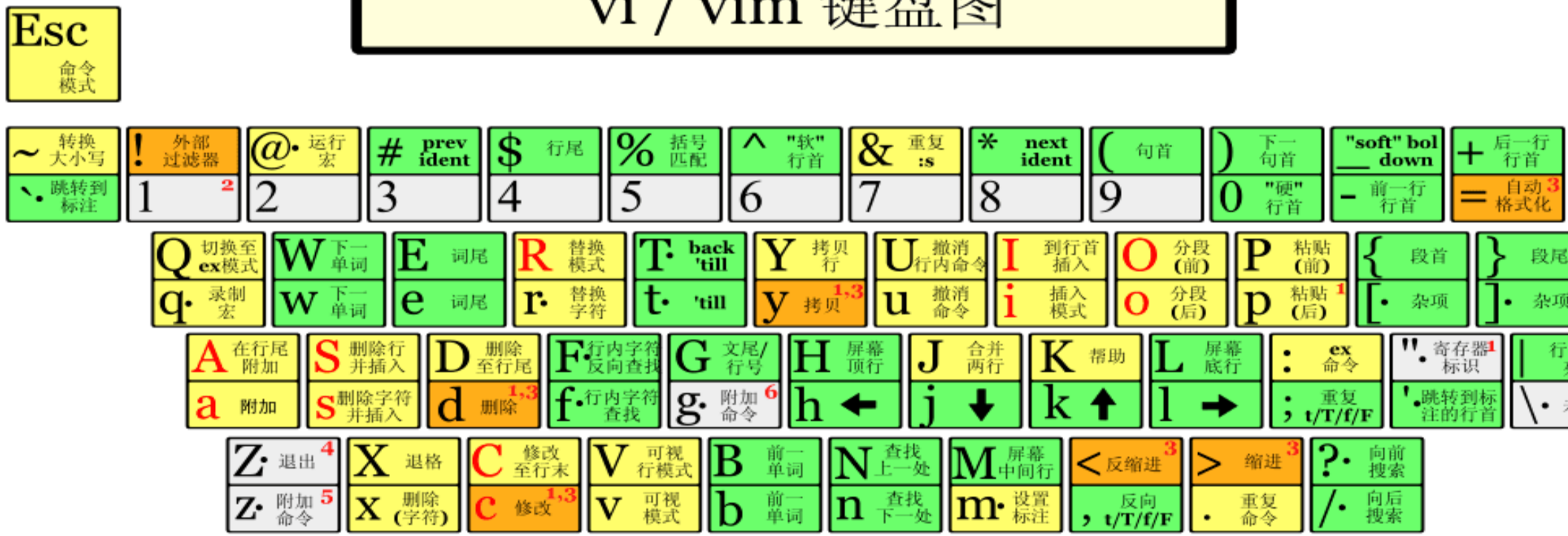
Movement of the Cursor

- `h` ←
- `j` ↓
- `k` ↑
- `l` →
- `0` moves the cursor to the beginning of the line.
- `$` moves the cursor to the end of the line.
- `w` moves forward one word.
- `b` moves backward one word.
- `G` moves to the end of the file.
- `gg` moves to the beginning of the file.

See https://www.runoob.com/linux/linux-vim.html for more details

version 1.1
April 1st, 06
翻译:2006-5-21

vi / vim 键盘图



动作 — 移动光标, 或者定义操作的范围

命令 — 直接执行的命令,
红色命令 进入编辑模式

操作 — 后面跟随表示操作范围的指令

extra — 特殊功能,
需要额外的输入

q· — 后跟字符参数

w,e,b命令

小写(b): `quux(foo, bar, baz);`
大写(B): `quux(foo, bar, baz);`

主要**ex**命令:
:w (保存), **:q** (退出), **:q!** (不保存退出)
:e f (打开文件 **f**),
:%s/x/y/g (**'y'** 全局替换 **'x'**),
:h (帮助 **in vim**), **:new** (新建文件 **in vim**),

其它重要命令:
CTRL-R: 重复 **(vim)**,
CTRL-F/-B: 上翻/下翻,
CTRL-E/-Y: 上滚/下滚,
CTRL-V: 块可视模式 **(vim only)**

可视模式:
漫游后对选中的区域执行操作 **(vim only)**

备注:
(1) 在 拷贝/粘贴/删除 命令前使用 **"x (x=a..z,*)**
使用命令的寄存器('剪贴板')
(如: **"ay$** 拷贝剩余的行内容至寄存器 **'a'**)
(2) 命令前添加数字
多遍重复操作
(e.g.: 2p, d2w, 5i, d4j)
(3) 重复本字符在光标所在行执行操作
(dd = 删除本行, **>>** = 行首缩进**)**
(4) **ZZ** 保存退出, **ZQ** 不保存退出
(5) **zt:** 移动光标所在行至屏幕顶端,
zb: 底端, **zz:** 中间
(6) **gg:** 文首 **(vim only)**,
gf: 打开光标处的文件名 **(vim only)**

原图: **www.viemu.com** 翻译: *fdl (linuxsir)*

VSCode

Visual Studio Code is a light weighed code editor.

https://code.visualstudio.com/Download

Recommendations

- The Missing Semester of Your CS Education
  - MIT course on how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more...
  - https://missing.csail.mit.edu/
- The Missing Semester lectures on bilibili
  - https://www.bilibili.com/video/BV1164y1u7mP