-Linux: a)cd b)cd ~ c)cd $home

Linux windows

-list content of.. ls /bin dir d:\

-In Linux, use rm -r to delete non-empty folders, rmdir to delete empty folders(windows same), and rm to delete files.

-In Windows, use rmdir /s to delete non-empty folders, and del to delete files.

-Touch/mkdir (windows/linux)

-cp 1.text 2.text(copy 1 into 2.text)

-r(递归复制）复制文件夹及其内容

---------------------WEEK3--------------------

# cat w1.txt w2.txt > combined.txt -> LINUX

copy /b w1.txt + w2.txt combined.txt ->WINDOWS

PWSH ↓

Get-Content w1.txt, w2.txt|Set-Content combined.txt

#less // 按需加载文件内容，支持上下滚动查看，适合查看大文件，不会消耗过多内存。

more // 从文件头开始按顺序读取显示，只能向下翻页，不支持回滚，适合快速查看文件前部分内容。

head // 默认显示文件的前10行，使用`-n`可指定行数。

head -n 5 example.txt // 显示前5行

head -c 5 example.txt // 显示前5个字节

tail //默认显示文件的最后10行,可使用`-n/-c。

# binery file后缀: .dat .bin(System executable)

# file 1.txt 检查文件类型 #linux

# cut -d',' -f1,5 uniweek5.txt > eg.txt

↑ 截取（不修改）源文件的第1,5列（以','为分隔) 并保存

((-d)elimiter)分隔符

((-f)ields区域

# sort /-n/-r/-u/-f eg.txt

↑ 默认按字母/-数值/-降序/-去重/-忽略大小写排序

# uniq /-d/-c eg.txt

↑ Remove duplicates (d: show duplicate lines, c: count occurrences)

# find /home -name "\*.txt"

↑ 在/hom目录包括子目录下查找所有扩展名为.txt 的文件

# grep -i "hello" example.txt文件中查找忽略大小写

-n 显示匹配行号 -r 递归查找

# diff -u eg1.txt eg2.txt 比较不同（-u统一格式）

# tr [选项] '原字符集' '目标字符集'

-d：删除指定的字符 -s：压缩重复的字符

# cat eg.txt | tr 'a-z' 'A-Z'

↑ 将源文件中的小写字母转换(tr)anslate)为大写字母并显示

# file globbing \*任意多/?任意单/[]集合单

# stat -c%s file.txt 查看目标信息

-c自定义输出格式 %s显示字节数

# mv/cp/rm/diff ->linux

Move/copy/del/fc ->windows

------------------WEEK4-------------------

# inode 是文件内容和元数据的实际存储位置

# ln 创建硬链接（Hard Link）用法=cp

↑ 共享 原文件inode/删除原文件时，其他硬链接依然有效

# ln -s 创建软链接（Symbolic Link）

↑ 指向文件路径，拥有自己的inode/删除原文件链接失效

# ls -i(check node)/-ld(查看目录信息不列出目录内容）

# find . -inum <inode\_number> 查找 inode 号为…的文件

# chmod u+rwx,go=rx file.txt

r=4, w=2, x=1 / octal: user-group-others

chmod 754 file.txt (binary:111/101/100)

# umask -default permission: dir(777) file(666)

Example: umask 077 means:

-Directory: 777- 077 = 700 rwx------

-File: 666 - 077 = 600 rw-------

umask设置会修改默认权限，作用于后续你创建的所有文件和目录, 直至关闭终端或者手动更改

--------------------WEEK5--------------------

# ps (静态显示当前终端进程）

pid:processing id/TTY:terminal type

pts: persudo terminal slave (like putty)

# ps -u $(whoami) == ps -u username （包含用户关联的守护进程daemon processes）

# ps -e :snap shot of current process@all terminal

# ps -l（和-u类似，都只显示当前节点，即user的进程）

# ps -p +进程号 （只显示该进程简单信息）

# pstree

# sleep 10 & （&:start the process in background)

ctrl+Z (stop process and send to background)

Ctrl+C 终止进程

bg：恢复为后台运行 fg: 恢复为前台运行 +作业号(jobs)

eg: fg %1 (1是作业号job number)

# command &> file

解释：&> 同时重定向标准输出和标准错误到同一个file

示例：ls &> output.txt，标准输出和标准错误都将写入…

# command > file 2> file

解释：>重定向标准输出到文件2>重定向标准错误到同一个文件

示例：ls > output.txt 2> error.txt，ls 的正常输出会写入output.txt，错误信息则写入 error.txt。

# kill -9 PID (or %job) 强制终止某PID

↑ -15清理资源后正常退出/-2中断信号=Ctrl+C/-19立即停止进程需要-18继续运行/-20终端停止信号=Ctrl+Z/-18继续执行,恢复被19SIGSTOP或20SIGTSTP暂停的进程继续运行

# stop-process -name notepad++ ->powershell

---------------------WEEK6---------------------

# top (动态显示所有终端进程)

-VIRT(virtual memory) 程序可使用总内存 = 正使用的物理ram + 未使用内存ram+ swap

-RES(resident set size) 实际使用物理内存

-SHR (shared memory)

shift+Press:'P'sort processes by CPU usage/'M'sort by memory usage/'N' sort by PID

# get-process ->top in powershell

----------------- WEEK8 & WEEK9------------------

# Directory Path Purpose

/ Root, top of file system hierarchy

/bin Common system binaries (commands)

/home Used to store users’ home directories

/usr This is call User, where user related programs live

/usr/bin Common utilities (commands) for users

/usr/sbin Common utilities for system admin

/etc System configuration files (eg. passwd)

/var Dynamic files (log and mail files)

/tmp , /var/tmp Temporary files for programs

/dev Device driver files (terminals, printers, etc)

/opt User installed application programs

# shebang-> #!/usr/bin/bash(或pwsh)

# get-childItem | format-list(pwsh)== ls -l (Linux)

set-executionPolicy remotesigned -scope currentuser

↑ pwsh需要添加签名证书到我的信任列表才能执行

-------------pwsh script----------------

$name="Alice"

global:$age=25

$fruits=@("apple","banana","cherry")

$greeting="Hello, "+$name

Write-Host $greeting

$sum=$age+5

write-output "In5years, $namewillbe $sum yearold."

$fruits|ForEach-Object{Write-Host "I like $\_."}

echo "PWSH Version: $($PSVersionTable.PSVersion)"

if($?) {Write-Host "The previous command was successful."} else {Write-Host "The previous command failed."}

$processName="chrome"

Get-Process -Name $processName

# $script:scriptVar="script-scoped!"作用整个脚本

# $local:Var="im local"作用域函数内

-------- Backslash:"\" ->bash/pwsh/cmd--------

backslash character outside of quote or inside double quotes tells the shell to ignore any special meaning that the following character may have.

----------environment variable----------------

# export Y=123:Set临时环境变量Y.子进程可用

# PS1="[\u@\h \W]\$ " Customize shell prompt.

↑ \u:Username \h:Hostname \W:Current directory

# PATH="$PATH:." Temporarily add current directory to PATH:cmd directory paths separated by‘:’

# LANG: Language/locale settings

# HOME: User's home directory.

# Exit Status Codes:

↑ When a program runs, it exits with a numeric value表示执行结果.执行一个命令后再 echo $?

0成功/1通用未知错误/2错误的命令或参数使用（misuse）

------------------#!/bin/bash-----------------

read -p "Enter your name: " name

echo "Hello, $name"

time=$(date) # $(command)指令替代:substitution

echo "Current time: $time"

x=$((3 + 2)) # $((…))求值并返回结果

((y = 4 \* 3)) # ((…))算术expression

echo "x=$x, y=$y"

----------------bash条件判断(if)-----------------

# if list1; then\nLIST1\nelif list2; then\nLIST2\n

Else\nLIST3\nfi

# if test "$NAME" == "Chris"; then

↑ 可作if的list指令,exit code=0判定为真

# if [[ -f myfile ]] #-f判定是否为普通文件

↑ -d:isDirectory/-l:isSymbolicLink/-r:isReadable

-w:isWritable/-x:isExecutable/-e:isExist

# $str1 == $str2 strings match/!= not match/> str1(sorts after)str2/< str1(sorts before)str2

# -z $str check if str empty/-n if not empty

# integer1 -eq integer2 areEqual/-ne notEqual/-gt greaterThan/-ge greaterThan or equal/-lt/-le

\*note: >和<用于字符串比较，整数比较使用运算符，如-gt等

# && || 与或---------script parameters----------

Bash脚本中 $0代表脚本路径./script name.

$1, $2, ...positional arguments:$1 first/$# 参数数量/shift :removes the 1st argument, shift rest forward/$\* 所有参数作单字符串/$@ 每个参数作独立字符串

---------for 遍历集合的值/C风格计数--------

for var in value1 value2 value3; do echo $var done

for (( i=0; i<5; i++ )); do echo $i done

# while/until [[ ]]; do list1 done

------------basic calculator(bc)-----------------

echo "3 + 5" | bc # Output: 8

echo "scale=2; 5/3" | bc # Output: 1.66

random\_float=$(echo "scale=2; $RANDOM/32767" | bc)

echo $random\_float # Output: 比如0.34

result=$(echo "scale=2; x=10; y=5; x/y" | bc)

echo $result # Output: 2.00

bc # Interactive Mode

# In interactive mode, you can enter commands like:

scale=2; 5/3 # Output: 1.66

----------------------WEEK11--------------------

# Break: When break executed, the loop stops and program moves to next line of code after the loop.

# Shared Librarie Reduce Memory Consumption/Storage Requirements/Independent Updates

# Common directories for executables:

↑ C:\Program FilesC:\Program Files (x86) C:\Windows\System C:\Windows\SysWOW64

# for libraries:C:\Program FilesC:\Program Files (x86)C:\Windows\SystemC:\Windows\SysWOW64

# headerfiles:C:\Program FilesC:\Program Files(x86)

# Executables and libraries are usually located in the same directory on Windows.

# Common directories for executables:/bin- /usr/bin-/sbin-/usr/sbin for shared libraries: /usr/lib-/usr/lib64 for header files:/usr/include

# Executables and libraries have their own directories on Linux.

----------------sudo apt command---------------

sudo

- Stands for "SuperUser Do"

- Grants temporary admin (root) privileges to execute commands

apt

- Advanced Package Tool

- A package management tool for Debian-based Linux distributions (e.g., Ubuntu)

# Common sudo apt commands

# 1. sudo apt update // 更新本地软件包列表（更新软件版本等）

# - Updates the local package database with the latest available software.

# - Does not install any software, just refreshes available software list.

# 2. sudo apt upgrade //升级已安装软件

# - Upgrades installed packages to the latest version available.

# 3. sudo apt install <package-name>

# - Installs the specified software package

# 4. sudo apt remove <package-name>

# - Uninstalls the specified software package.

# 5. sudo apt purge <package-name>

# - Removes the software package along with its configuration files.

# 6. sudo apt autoremove

# - Removes unnecessary dependencies to save system space.

# 7. sudo apt show <package-name>

# - Displays detailed information about the software package (e.g., version, description).

# 8. apt list --installed

# - Lists all packages currently installed on the system.

# Program to Hardware Interaction

↑ Program -> Library Functions -APIs-> Operating System (Kernel)-> Hardware.

--------------IFS 和 read -r -------------------

1. IFS（Internal Field Separator）

- 描述：IFS是Bash中的一个特殊变量，用于定义分隔符，用于分割输入的数据。默认包含空格、制表符和换行符。

- 示例：

IFS=',' read -r name age

2. read 命令

- 描述：read命令用于从标准输入读取一行数据，并将其拆分为多个变量。

- `-r` 选项：防止反斜杠（\）被转义。

- 示例：

read -r first\_name last\_name

3. done < 用法

- 描述：done < 将文件内容传递给 while 循环中的 read 命令，逐行读取文件内容。

- 示例：

while IFS=',' read -r name age; do

echo "Name: $name, Age: $age"

done < "students.txt"

4. 综合示例

- 示例：读取 CSV 文件并处理每行数据。

IFS=',' read -r name age grade

while IFS=',' read -r name age grade; do

echo "Name: $name, Age: $age, Grade: $grade"

done < "students.csv"

----------# WINGET------#apt -----------------

winget search/list/show/install/uninstall package

sudo apt update更新本地索引/upgrade升级软件/install /purge/show nginx

apt list --installed

------------------------WEEK12------------------

1) #grep

# Use -i option to ignore case

# Recursive search: searches in all files under the current directory: grep -r 'need' .

# Show line numbers: grep -n 'error' logfile.txt

# Exact word match: "error" but not "errors" or "ERRORS": grep -w 'error' logfile.txt

1.1) Regular Expressions

# Matches "Friend" or "friend" only

↑ egrep '[Ff]riend' poem2.txt

# Matches "Friend", "friend", "Ffriend", "fFriend"

↑ egrep '[Ff]+riend' poem2.txt

---------Components of Regular Expressions-------

# Atoms

- Single character: a, 1, $

Example: grep 'a' file.txt (matches any line containing 'a')

- Escape character: \d (matches a digit)

Example: egrep '\d' file.txt (matches any line containing a digit)

========# Wildcards

- . : Matches any single character

Example: egrep 'a.b' file.txt (matches 'aab', 'acb', etc., but not 'ab')

=========# Character classes

- []: Defines a set of characters

Example: egrep '[aeiou]' file.txt (matches any vowel)

- [a-z]: Matches a range

Example: egrep '[a-z]' file.txt (matches any lowercase letter)

- ^ inside []: Negates a set

Example: egrep '[^0-9]' file.txt (matches any non-digit)

=========# Repetition

- .....: Matches exactly 5 consecutive characters.

Equivalent to `.{5}`

- + : Matches one or more preceding atom

Example: egrep 'go+' file.txt (matches 'go', 'goo', 'gooo', etc.)

- \* : Matches zero or more preceding atom

Example: egrep 'go\*' file.txt (matches 'g', 'go', 'goo', etc.)

- ? : Matches zero or one preceding atom

Example: egrep 'colou?r' file.txt (matches 'color' or 'colour')

- {n}: Matches exactly n preceding atom

Example: egrep 'a{3}' file.txt (matches 'aaa')

- {n,}: Matches at least n preceding atom

Example: egrep 'a{2,}' file.txt (matches 'aa', 'aaa', 'aaaa', etc.)

- {n,m}: Matches between n and m preceding atom

Example: egrep 'a{2,4}' file.txt (matches 'aa', 'aaa', 'aaaa')

# Important notes #

- Repetition qualifiers only for preceding element.

- Use () group multiple characters: (ab){3}.

\* Repetition qualifiers are placed after the character or group they modify.

\* Single character repetition

a{3} # Matches "aaa" (exactly three consecutive 'a's).

\* Group repetition

(ab){3} # Matches "ababab" (three consecutive "ab" groups).

\* Wildcard repetition

.\* # Matches any character (except newline) zero or more times.

\* Character class repetition

[a-z]{2,4} # Matches 2 to 4 lowercase letters.

========# Alternation

- | : Specifies alternatives

Example: egrep 'cat|dog' file.txt (matches 'cat' or 'dog')

=========# Grouping

- (): Groups expressions

Example: egrep '(ab)+' file.txt (matches 'ab', 'abab', 'ababab', etc.)

=========# Capturing/Backreferences

- (): Captures matched groups

Example: egrep '(go)+od' file.txt (matches 'good', 'gogood', 'gogogood')

- \n: Refers to nth captured group

Example: egrep '([a-z])\1' file.txt (matches repeated characters like 'aa' or 'bb')

=========2.1) # Examples and Explanations

# Match str start with a digit followed by letters:

egrep '^\d[a-zA-Z]+$' file.txt

- `^`: Start of the string.

- `\d`: A single digit (0-9).

- `[a-zA-Z]+`: >=1 letters (upper or lowercase).

- `$`: End of the string.

Matches: `1abc`, `7Hello`.

Non-matches: `abc1`, `123`.

# Match an email address:

egrep '^\w+@\w+\.\w+$' file.txt

- `^`: Start of the string.

- `\w+`: One or more word characters (letters, digits, `\_`).

- `@`: Literal `@` character.

- `\w+`: One or more word characters for the domain.

- `\.`: A literal dot.

- `\w+`: One or more word characters for the top-level domain.

- `$`: End of the string.

Matches: `user@gmail.com`, `name123@domain.org`.

Non-matches: `@gmail.com`, `user@domain`.

# Match an IP address:

egrep '^((25[0-5]|2[0-4]\d|[01]?\d\d?)\.){3}(25[0-5]|2[0-4]\d|[01]?\d\d?)$' file.txt

- `^`: Start of the string.

- `(25[0-5]|2[0-4]\d|[01]?\d\d?)`: Match 0-255:

- `25[0-5]`: Numbers 250-255.

- `2[0-4]\d`: Numbers 200-249.

- `[01]?\d\d?`: Numbers 0-199 (optional leading `0` or `1`).

- `\.`: Literal dot.

- `{3}`: Repeat 3 times for 4 total segments.

- `$`: End of the string.

Matches: `192.168.0.1`, `255.255.255.255`.

Non-matches: `256.1.1.1`, `192.168.1`.

=======

2.2) # More Rules & Anchors

- '\.' Matches lines with a literal period.

- '^start' Anchors at the start of a line.

- 'end$' Anchors at the end of a line.

- '\<te' Matches 'te' at the start of a word.(like 'test')

- 'ld\>' Matches 'ld' at the end of a word.(like 'world')

# MM/DD/YYYY or MM-DD-YYYY

egrep '\b(0[1-9]|1[0-2])[/-](0[1-9]|[12][0-9]|3[01])[/-][0-9]{4}\b' sample.txt

# email egrep '\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b' sample.txt

# URL egrep 'https?://[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}(/\S\*)?' sample.txt

---------------Windows cmd---------------

REM等价于:: Variable Assignment ↓

SET VAR\_NAME=value :: Assign value to a variable

ECHO %VAR\_NAME% :: Access variable value

:: Read User Input ↓

SET /P NAME=Enter your name:

ECHO Hello, %NAME%! :: Example of user input

:: Clear Variable ↓

SET VAR\_NAME= :: Clear VAR\_NAME

:: String Replacement ↓

SET TEXT=Welcome to Windows

ECHO %TEXT:Windows=CMD% ::Replace"Windows"with"CMD"

:: Substring Extraction ↓

SET FILE=report.txt

ECHO %FILE:~0,6% ::Output:"report"(first 6chars)

:: Environment Variables ↓

SET :: List all environment variables

ECHO %PATH% :: Display PATH variable

SET PATH=%PATH%;C:\NewPath :: Append to PATH

:: Persistent Variables ↓

SETX VAR\_NAME value :: Permanently set VAR\_NAME

-----------------powershell-----------------

# Variables ↓

$Var = value # Assign value

$Num = 10 # Number

$Text = "Hello" # String

$Bool = $true # Boolean

# Expressions ↓

$Sum = 5 + 3 # Result: 8

$Concat = "Power" + "Shell" # Result: PowerShell

# Command output to variable ↓

$Date = Get-Date

$Processes = Get-Process

# Multiline string ↓

$MultiText = @"

Line 1

Line 2

"@

# Arrays ↓

$Array = 1, 2, 3

$Array += 4 # Add 4

$Array[0] # Access 1st element

# Hash tables ↓

$HashTable = @{Key1="Value1"; Key2="Value2"}

$HashTable["Key1"] # Get Value1

# String substitution ↓

$Name = "Alice"

$Greeting = "Hello, $Name!" # Result: Hello, Alice!

# Output ↓

Write-Output "Hello, $Name!" # Print: Hello, Alice!

Write-Host "Hello, $Name!" # Same as above

"Hello, $Name!" # Simplified output

# Environment variables ↓

$env:PATH += ";C:\NewPath" # Add to PATH