watch -n 1 "/sbin/ifconfig eno16777736 | grep bytes"

top -p `pgrep python | tr "\\n" "," | sed 's/,$//'`

nohup:

nohup command > myout.file 2>&1 &

使用 jobs 查看任务

使用 fg %n关闭

清理缓存：

echo 3 > /proc/sys/vm/drop\_caches

crontab :

10 \* \* \* \* root bash /home/sparkwork/timer\_monitor\_v1.0.1/scheduler\_timer\_crontab.sh >> /home/sparkwork/timer\_monitor\_v1.0.1/logs/scheduler\_monitor.log 2>1&

**从根目录开始查找所有扩展名为.log的文本文件，并找出包含”ERROR”的行  
find / -type f -name "\*.log" | xargs grep "ERROR"  
例子：从当前目录开始查找所有扩展名为.in的文本文件，并找出包含”thermcontact”的行****find . -name "\*.in" | xargs grep "thermcontact"**

**wc命令**

该命令各选项含义如下：

　　- c 统计字节数。

　　- l 统计行数。

　　- w 统计字数。

$ wc - lcw file1

1.统计demo目录下，js文件数量：

find demo/ -name "\*.js" |wc -l

2.统计demo目录下所有js文件代码行数：

find demo/ -name "\*.js" |xargs cat|wc -l 或 wc -l `find ./ -name "\*.js"`|tail -n1

3.统计demo目录下所有js文件代码行数，过滤了空行：

find /demo -name "\*.js" |xargs cat|grep -v ^$|wc -l

wget:

有时候我们需要[wget](http://java-er.com/blog/tag/wget/" \t "_blank" \o "查看 wget 中的全部文章)一个文件下载到指定的目录下，或者重命名成指定的名字  
[wget](http://java-er.com/blog/tag/wget/) -r -p -np -k -P ~/tmp/ <http://java-er.com>

解释一下参数

-P 表示下载到哪个目录

-r 表示递归下载

-np 表示不下载旁站连接.不下载父目录

-k 表示将下载的网页里的链接修改为本地链接.

-p 获得所有显示网页所需的元素

额外的

-c 断点续传

-nd 递归下载时不创建一层一层的目录，把所有的文件下载到当前目录

-L 递归时不进入其它主机，如wget -c -r www.xxx.org/

-A 指定要下载的文件样式列表，多个样式用逗号分隔

-i 后面跟一个文件，文件内指明要下载的URL

\* 使用代理下载

wget -Y on -p -k https://sourceforge.net/projects/wvware/

代理可以在环境变量或wgetrc文件中设定

tar解压：

1)对于.tar结尾的文件   
　　tar -xf all.tar   
　　2)对于.gz结尾的文件   
　　gzip -d all.gz   
　　gunzip all.gz   
　　3)对于.tgz或.tar.gz结尾的文件   
　　tar -xzf all.tar.gz   
　　tar -xzf all.tgz   
　　4)对于.bz2结尾的文件   
　　bzip2 -d all.bz2   
　　bunzip2 all.bz2   
　　5)对于tar.bz2结尾的文件   
　　tar -xjf all.tar.bz2   
　　6)对于.Z结尾的文件   
　　uncompress all.Z   
　　7)对于.tar.Z结尾的文件   
　　tar -xZf all.tar.z

8)Windows下的zip

unzip all.zip

回收站功能在Linux中的实现：

将下列代码一次保存

#Save as /bin/delete

#!/bin/bash

realrm="/bin/rm"

if [ ! -d ~/trash ]

then

mkdir -v ~/trash

chmod 777 ~/trash

fi

if [ $# -eq 0 ]

then

echo "Usage: delete file1 [file2 file3....]"

echo "If the options contain -f, then the script will exec 'rm' directly"

fi

while getopts "dfiPRrvw" opt

do

case $opt in

f)

exec $realrm "$@"

;;

\*)

# do nothing

;;

esac

done

echo -ne "Are you sure you want to move the files to the trash?[Y/N]:\a"

read reply

if [ $reply = "y" -o $reply = "Y" ]

then #####

for file in $@

do

if [ -f "$file" -o -d "$file" ]

then

if [ -f "$file" ] && [ `ls -l $file|awk ' {print $5}'` -gt 2147483648 ]

then

echo "$file size is larger than 2G, will be deleted directly"

`rm -rf $file`

elif [ -d "$file" ] && [ `du -sb $file|awk '{print $1}'` -gt 2147483648]

then

echo "The directory:$file is larger than 2G, will be deleted directly"

`rm -rf $file`

fi

fi

done

fi

now=`date +%Y%m%d\_%H\_%M\_%S`

filename="${file##\*/}"

newfilename="${file##\*/}\_${now}"

mark1="."

mark2="/"

if [ "$file" = ${file/$mark2} ]

then

fullpath="$(pwd)/$file"

elif [ "$file" != ${file/$mark1} ]

then

fullpath="$(pwd)${file/$mark1}"

else

fullpath="$file"

fi

echo "the full path of this file is : $fullpath"

if mv -f $file ~/trash/$newfilename

then

$(/bin/logTrashDir "$newfilename $filename $now $fullpath")

echo "files: $file is deleted"

else

echo "the operation is failed"

fi

#Save as /bin/logTrashDir

#!/bin/bash

if [ ! -f ~/trash/.log ]

then

touch ~/trash/.log

chmod 700 ~/trash/.log

fi

echo $1 $2 $3 $4>> /home/razrlele/trash/.log

#Save as /bin/restoreTrash

#!/bin/bash

originalPath=$(awk /$filename/'{print $4}' "$HOME/trash/.log")

filenameNow=$(awk /$filename/'{print $1}' "$HOME/trash/.log")

filenamebefore=$(awk /$filename/'{print $2}' "$HOME/trash/.log")

echo "you are about to restore $filenameNow,original name is $filenamebefore"

echo "original path is $originalPath"

echo "Are you sure to do that?[Y/N]"

read reply

if [ $reply = "y" ] || [ $reply = "Y" ]

then

$(mv -b "$HOME/trash/$filenameNow" $originalPath)

$(sed -i /$filenameNow/'d' "$HOME/trash/.log")

else

echo "no files restored"

fi

#Save as /bin/cleanTrashCan

#!/bin/bash

arrayA=($(find ~/trash/\* -mtime +7 | awk '{print $1}'))

for file in ${arrayA[@]}

do

$(rm -rf "${file}")

filename="${file##\*/}"

echo $filename

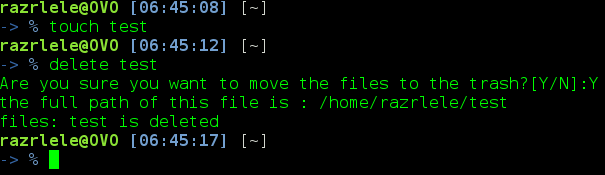
$(sed -i /$filename/'d' "$HOME/trash/.log")

done

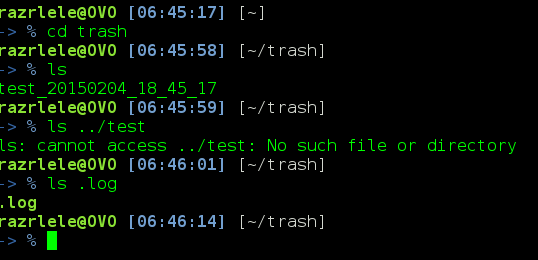
赋予脚本执行权限:

chmod +x delete restoreTrash logTrashDir cleanTrashCan

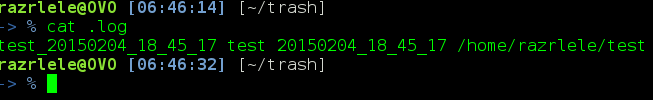
delete脚本执行过程：



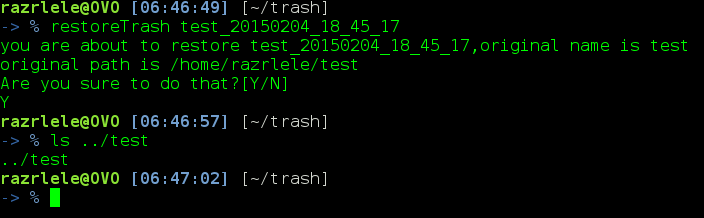
回收站目录：



生成.log记录：



restoreTrash脚本执行过程：



最后再用system/Timers定期没星期日下午六点执行cleanTrashCan脚本，首先编辑

# /etc/systemd/system/cleanTrash.service

[Unit]

Description=Clean Trash Can

[Service]

Type=simple

ExecStart=/bin/cleanTrashCan

再编辑

# /etc/systemd/system/cleanTrash.timer

[Unit]

Description=Runs cleanTrashCan every week

[Timer]

OnCalendar=Sun, 18:00

Unit=cleanTrash.service

[Install]

WantedBy=multi-user.target

执行：

# systemctl start cleanTrash.timer

# systemctl enable cleanTrash.timer

References:

https://www.ibm.com/developerworks/cn/linux/1410\_licy\_linuxtrash/

https://wiki.archlinux.org/index.php/Systemd/Timers

bundle install 命令：

Bundle介绍：

Rails 3中引入Bundle来管理项目中所有gem依赖，该命令只能在一个含有Gemfile的目录下执行，如rails 3项目的根目录。

关于Gemfile和Gemfile.lock

所有Ruby项目的信赖包都在Gemfile中进行配置，不再像以往那样，通过require来查找。Rails 3中如果需要require某个gem包，必须通过修改Gemfile文件来管理。

Gemfile.lock则用来记录本机目前所有依赖的Ruby Gems及其版本。所以强烈建议将该文件放入版本控制器，从而保证大家基于同一环境下工作。

Bundle命令详解：

# 显示所有的依赖包

$ bundle show

# 显示指定gem包的安装位置

$ bundle show [gemname]

# 检查系统中缺少那些项目以来的gem包

# 注：如果系统中存在所有项目以来的包，则会输出：The Gemfile's dependencies are satisfied

$ bundle check

# 安装项目依赖的所有gem包

# 注：此命令会尝试更新系统中已存在的gem包

$ bundle install

# 安装指定的gem包

$ bundle install [gemname]

# 更新系统中存在的项目依赖包，并同时更新项目Gemfile.lock文件

$ bundle update

# 更新系统中指定的gem包信息，并同时更新项目Gemfile.lock中指定的包信息

$ bundle update [gemname]

# 向项目中添加新的gem包引用

$ gem [gemname], [ver]

# 你还可以指定包依赖关系

$ gem [gemname], :require => [dependence\_gemname]

# 你甚至还可以指定gem包的git源

$ gem [gemname], :git => [git\_source\_url]

# 锁定当前环境

# 可以使用bundle lock来锁定当前环境，这样便不能通过bundle update来更新依赖包的版本，保证了统一的环境

$ bundle lock

# 解除锁定

$ bundle unlock

# 打包当装环境

# bundle package会把当前所有信赖的包都放到 ./vendor/cache/ 目录下，发布时可用来保证包版本的一致性。

$ bundle package

$ bundle install [--binstubs=PATH] [--clean] [--deployment] [--frozen]

[--full-index] [--gemfile=FILE] [--local] [--no-cache]

[--no-prune] [--path=PATH] [--quiet] [--shebang=STRING]

[--standalone=ARRAY] [--system] [--without=GROUP GROUP]

[--trust-policy=SECURITYLEVEL]

Options:

--binstubs: Generate bin stubs for bundled gems to ./bin

--clean: Run bundle clean automatically after install

--deployment: Install using defaults tuned for deployment environments

--frozen: Do not allow the Gemfile.lock to be updated after this install

--full-index: Use the rubygems modern index instead of the API endpoint

--gemfile: Use the specified gemfile instead of Gemfile

--jobs: Install gems using parallel workers.

--local: Do not attempt to fetch gems remotely and use the gem cache instead

--no-cache: Don't update the existing gem cache.

--no-prune: Don't remove stale gems from the cache.

--path: Specify a different path than the system default ($BUNDLE\_PATH or $GEM\_HOME). Bundler will remember this value for future installs on this machine

--quiet: Only output warnings and errors.

--retry: Retry network and git requests that have failed.

--shebang: Specify a different shebang executable name than the default (usually 'ruby')

--standalone: Make a bundle that can work without the Bundler runtime

--system: Install to the system location ($BUNDLE\_PATH or $GEM\_HOME) even if the bundle was previously installed somewhere else for this application

--trust-policy: Sets level of security when dealing with signed gems. Accepts `LowSecurity`, `MediumSecurity` and `HighSecurity` as values.

--without: Exclude gems that are part of the specified named group.

Gems will be installed to your default system location for gems. If your system gems are stored in a root-owned location (such as in Mac OSX), bundle will ask for your root password to install them there.

While installing gems, Bundler will check vendor/cache and then your system's gems. If a gem isn't cached or installed, Bundler will try to install it from the sources you have declared in your Gemfile.

The --system option is the default. Pass it to switch back after using the --path option as described below.

Install your dependencies, even gems that are already installed to your system gems, to a location other than your system's gem repository. In this case, install them to vendor/bundle.

$ bundle install --path vendor/bundle

Further bundle commands or calls to Bundler.setup or Bundler.require will remember this location.

Install all dependencies except those in groups that are explicitly excluded.

$ bundle install --without development test

Install all dependencies on to a production server. Do not use this flag on a development machine.

$ bundle install --deployment

The --deployment flag activates a number of deployment-friendly conventions:

Isolate all gems into vendor/bundle

Require an up-to-date Gemfile.lock

If bundle package was run, do not fetch gems from rubygems.org. Instead, only use gems in the checked in vendor/cache

Install gems parallely by starting the number of workers specificed.

$ bundle install --jobs 4

Retry failed network or git requests.

$ bundle install --retry 3

自己的：