Linux 系统自动化运维管理平台实现及应用*

黄建桥　　林宁**

南宁学院信息工程学院，南宁 530200

摘　要　为了提高 IT 运维工作效率，帮助企业能够主动对服务器进行监控管理，降低企业的运维成本，本文设计实现了一套自动化运维管理系统。该系统以 Ansible 技术作为主要自动化运维工具，采用红帽提供的带可视化界面的 Ansible Tower 自动化管理平台作为本系统的 Web 管理平台。系统主要由自动化检测监控模块、告警信息模块、跳板机模块、系统在线管理模块等组成，通过自动化检测监控脚本，对故障节点和故障信息可以进行及时告警通知运维人员，促使运维人员主动的、及时的、有效率的了解系统情况，并在线远程对服务器作出相应处理，从而提高运维工作效率，减轻工作负担。

关键字　系统运维，自动化，Ansible，Shell

Realization and Application of Automatic Operation and Maintenance Management Platform of Linux System

Huang Jianqiao　Lin Ning

School of Information Engineering Nanning University
Nanning 530200, China

Abstract— In order to improve the efficiency of IT operation and maintenance, help enterprises to proactively monitor and manage servers, and reduce the operation and maintenance costs of enterprises, a set of automatic operation and maintenance management system is developed. This system uses Ansible technology as the main automatic operation and maintenance tool, and utilizes the Ansible Tower automation management platform with visual interface provided by Red Hat as the Web management platform of this system. It mainly consists of automatic detection and monitoring module, alarm information module, springboard machine module, and online system management module. Through automatic detection and monitoring script, the fault node and fault information can be notified to the operation and maintenance personnel in time, so that the operation and maintenance personnel can proactively, timely and efficiently understand the system situation, and make corresponding online and remote processing of the server. Thus improve the operation and maintenance work efficiency, reduce the burden of work.

Keyword—System operation and maintenance, Automation, Ansible , Shell

1　引　言

随着信息技术的高速发展和信息化建设的不断完善，信息化产品也在逐渐地产生和普及到日常生活中，这些信息化产品在使用的过程中都需要运行维护。因此，信息化产品的运维已经成为信息服务的重要组成部分之一[1]。

面对越来越复杂的信息化业务和多样化的用户需求，需要越来越多的 IT 运维人员来保障系统的稳定运行。更需要灵便捷活的运维管理方式来进行有效保障。而传统的运维管理方式主要是依靠人工巡检的方式来完成对设备的管理[2]。当存在大规模的设备需要管理维护时，显然单靠人工巡检的方式，通过简单的表格记录，存在工作效率低、工作内容繁琐、人员方面要求高且容错率低，已经无法满足在技术、业务、管理等方面的要求。目前仍有很大一部分的信息管理系统，其运维方式依然停留在人工操作处理的层次，需要运维人员逐一地对系统内的设备进行手动管理和维护，在系统内设备规模较大时，无疑将会造成巨大的工作压力[3]。对系统的日常维护每天都要进行，这种重复性、低效率的手动操作模式将导致极大的人力与物力资源损耗，无形中加大了企业对信息管理系统的运维成本，并且手动完成的运维效果无法得到保障[4]。

技术发展带来的问题必须用技术来解决，只有在运维管理领域引入新技术、新理念，才能更好地提高运维管理水平[5]。针对以上问题，本文基于 Ansible 框架作为自动化运维工具，结合 Shell 脚本，实现了一个能为企业提高工作效率，降低运维成本的自动化运维

*基金项目：本文得到广西高校中青年教师科研基础能力提升项目（2021KY1805）资助。
**通讯作者：林宁，副教授，bgy_2009@163.com.

管理系统，帮助企业监控和管理服务器，确保系统的正常运转。

2 相关技术研究

2.1 Ansible 框架

Ansible 是以 Python 为基础开发的一个开源运维自动化工具，它集结了多种自动化运维管理工具的特点，具有批量系统配置管理、批量应用部署、任务编排等多种功能，通过 ssh 协议与远程主机连接，不需要在被管理节点上安装额外的软件，就可以对系统进行远程管理，批量自动化执行命令脚本，支持多节点发布，远程执行任务等。

Ansible 的特点有：安装部署很容易，仅需将 Ansible 环境安装到主控端服务器上，受控端服务器无需安装任何客户端软件；具有很强大的功能且配置较简单，扩展性强；具有大量内置模块，通过调用模块来完成特定的任务；默认使用 ssh 协议与受控端连接通信；可以批量远程执行命令和脚本。

Ansible 是基于模块的调用来进行工作的，通过模块的运行，实现批量部署的能力。由 6 个部分组成：Ansible、Ansible Playbook、Host Inventory、Modules、Connection Plugins 和 API。其中，Ansible 是核心执行工具，负责发放命令；Ansible Playbook 被称为任务剧本，属于 YAML 格式，依据 Ansible 发布的命令来定义主机需要调用哪些模块来完成相关的任务[6]。Host Inventory 是主机清单文件，记录着主机的端口号、IP 地址信息等；Modules 是一个模块，用来执行命令；Connection Plugins 为连接插件，用来连接受控端进行通信；API 为程序调用的物理接口。以上 6 个部分组成一个架构，如图 1 所示。

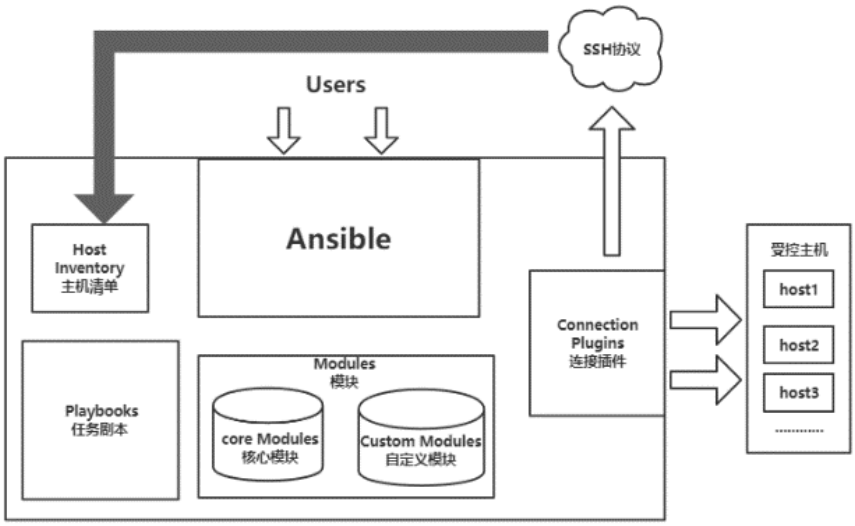

图 1 Ansible 架构图

2.2 Ansible Tower

Ansible Tower 是 Redhat 基于 Django 框架编写的，为 Ansible 提供的一个前端 UI，是一个基于 web 的超级强大的自动化集中管理平台。通过将 Ansible 的指令进行界面化后，拥有操作简明直观，简单易用的特点。

Ansible Tower 向运维管理人员提供一个网页接口，通访问该接口就进入到前端管理平台，在该平台上，管理员可以管理主机清单，共享所有的 Playbooks 脚本给主机清单里的主机使用，在平台上执行 Playbooks 任务局本后，Ansible 会通过 ssh 协议连接受控端主机后进行任务剧本的执行，根据任务剧本的执行结果，可以更好的分析目标主机的状态。

2.3 Linux Shell

在 Linux 操作系统中，用户所输入的指令是不能被识别的，所以这就必须要有一个程序来协助进行转换，使之成为系统能够辨识的指令，然后再将产生的结果返回给用户。提供转换功能的程序就是命令解释器 Shell，Shell 是位于操作系统最外层的指令解释者，它的任务是负责与用户直接进行交流，将用户输入的信息解释给操作系统，并对不同操作系统的输出结果进行处理，最终将处理结果输出至屏幕提供给用户使用，该会话方式有交互式和非交互式。

Shell 不仅是 Linux 操作系统中的一个命令语言

也是程序设计语言。将多个 Shell 和操作系统命令按照语法结合起来，然后把它们存储在一个扩展名为 sh 的文本文件里，就形成一个 Shell 脚本，也叫 Shell Script 脚本，该脚本能够很容易地与系统进行交互，完成对系统的管理和批量操作。

2.4 SSH 协议

SSH（Secure Shell）是由 IETF 网络工作小组制定的一个网络通信安全协议[7]。该协议的主要功能是为远程连接服务器提供的一个安全协议，并且在服务器操作系统中提供默认使用的端口号。在通信的过程中，所有的数据都是经过加密的，为数据传输的安全性提供了保障。

SSH 的安全验证方式有：

第一个安全验证方式是使用口令密码，用户已知服务器的账号和密码后就能使用 SSH 协议进行远程连接服务器，并且 SSH 是基于 TCP 协议的[8]。因此数据在传输的过程中具有一定的安全性。

第二个安全验证方式是使用密钥[9]。该密钥是通过加密算法进行加密后得到的，客户端首先要生成本地的一组密钥，再将公钥部分通过通信协议将数据发送至服务器端，服务器端对密钥进行加密处理，通过该验证方式有效的提高了 SSH 网络通信协议的安全性，在连接时能够有效的抵御外来的网络攻击和入侵。

3 系统总体设计

在本系统的总体设计中，采用了模块划分的方式，该方式便于后期功能的添加、修改和维护[10]。

本系统的 web 管理平台部分采用的是由红帽提供的 Ansible Tower 集中管理平台，已经含有一些基本的命令批量部署执行的功能。针对未实现的功能，本文通过两部分进行总体的设计，有前台检测部分和后台管理部分。前台检测是指在 web 平台上就能实现的功能，后台则是需要登录到服务器系统内部进行操作。其中前台检测部分包括自动化检测监控模块和告警信息模块设计，后台管理部分包括跳板机模块和系统在线管理模块两部分。

3.1 自动化检测监控模块设计

自动化检测监控模块主要分为以下子模块：检测 CPU 利用率、检测内存利用率、检测磁盘空间利用率、检测磁盘 I/O 性能、检测网络性能、检测系统服务运行状态、检测系统端口占用情况、检测系统负载、检测系统进程运行情况、检测系统僵死进程、检测主机存活状态、检测 TCP 连接数量这十二个功能模块。

3.2 告警信息模块设计

在运维工作中，监控告警是不可缺少的一部分，对服务器系统发现的故障第一时间进行及时告警通知，提高运维人员处理故障的响应时间，从而保障服务器系统的稳定运行。

告警信息模块由两部分组成，分别是告警内容和通知方式。其中告警内容包含故障服务器的：主机名、主机 IP、模块检测结果、故障信息、检测时间等，通知方式则是通过 QQ 邮箱通知的方式。告警信息模块如图 2 所示。

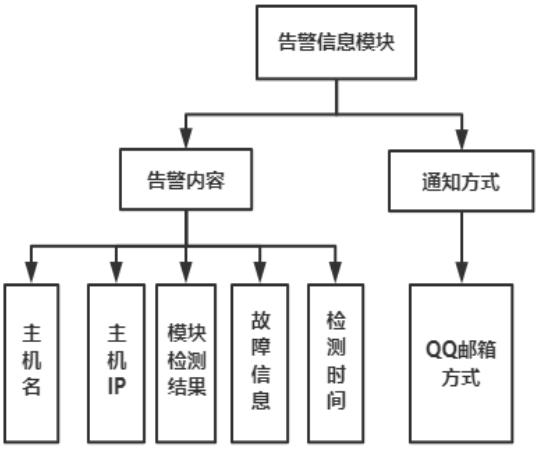

图 2 告警信息模块

3.3 跳板机模块设计

跳板机作为一台用来访问的服务器，须通过这台服务器才能去访问别的服务器。跳板机和服务器之间采用 SSH 协议进行连接，并且限制所有服务器只能通过跳板机访问，仅开放 SSH 远程协议的 22 端口给跳板机使用，这种访问方式减少了端口的暴露，可以在一定程度上提高服务器的安全性。

3.4 系统在线管理模块设计

本模块可以帮助运维人员良好的对故障节点发生的故障信息进行修复处理。当运维人员接收到故障告警信息后，登录故障节点修复故障时，为其提供的一个可以快速解决故障问题的模块。本模块具有友好的中文语言交互界面，简单便捷灵活的操作，无需运维人员输入任何操作命令即可修复故障，可以有效避免运维人员对系统操作命令的局限性。本模块主要分为以下子模块：用户管理模块、服务管理模块、防火墙管理模块、系统进程管理模块、数据库管理模块、查看检测模块日志文件这六个功能模块。

4 系统实现

在完成了系统的总体设计后，为了保证系统的正常运行，首先要进行初步的工作，那就是部署好基本的运行环境，主要包括 Python 语言环境的安装和 SSH 远程服务工具的开启和运行。

在 Linux Centos7 操作系统下，默认已经安装好了 Python 的 2.7 版本 Openssh 工具的 7.4 版本，并且也默认启动了 Openssh 服务，如未启动，则需通过“systemctl start sshd”命令启动。

4.1 自动化检测监控模块

为了确保自动化检测监控模块程序的自动后台运行，运用到了 CentOS7 操作系统中的一个系统定时器“Crontab”，Crontab 用于提交和管理执行周期性的任务，因此在 Crontab 里设定了监控程序的检查时间，以便定期地执行监控程序。监控程序会分别在每一个小时的第 5 分钟、20 分钟、35 分钟、50 分钟自动执行，确保监控程序在一个小时内运行四次，达到自动化检测监控的效果。

（1） 检测 CPU 利用率模块

在检测 CPU 利用率模块中，实现步骤分为以下五步。

第一步，通过“iostat –c”命令输出 CPU 使用情况，结合 awk 工具进行相关参数的取值后，根据取值计算 CPU 利用率。

第二步，设定监控阈值。

第三步，做一个判断，判断第 CPU 的利用率是否超过监控阈值。第一种情况，未超过监控阈值，输出当前系统的 CPU 使用情况。第二种情况，超过监控阈值，输出当前系统的 CPU 使用情况和故障原因，并发送告警邮件通知。

第四步，输出占用 CPU 最高的前五个进程。

第五步，将检测结果记录到日志文件中。

检测 CPU 利用率模块流程图如图 3 所示。

该功能核心代码如下：

```
user=`iostat -c |awk 'NR==4 {print $1}'`
sys=`iostat -c |awk 'NR==4 {print $3}'`
iowait=`iostat -c |awk 'NR==4 {print $4}'`
idle=`iostat -c |awk 'NR==4 {print $NF}'`
util=`echo
"scale=2;a=($user+$sys+$iowait);if(length(a)==scale(a))        print
0;print a" |bc`
#监控阈值
max=50.00
#用 bc 对小数比较  true：1 false:0
bc=`echo "$util>$max"|bc`
```

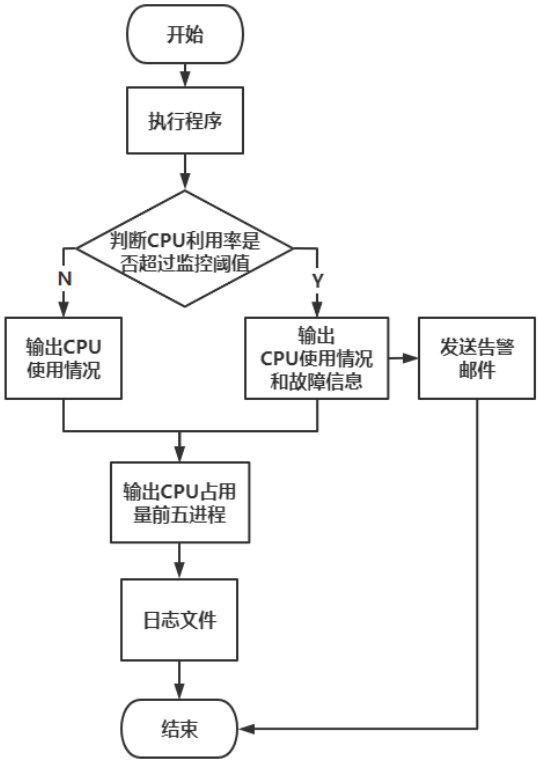

图 3 检测 CPU 利用率流程图

（2） 检测内存利用率模块

检测内存利用率模块，实现步骤分为以下五步。

第一步，通过“free -m”命令输出内存的使用情况，结合 grep、awk 工具进行相关参数的取值，根据取值计算内存利用率。

第二步，设定监控阈值。

第三步，做一个判断，判断内存的利用率是否超过监控阈值。第一种情况，未超过监控阈值，输出当前系统的内存使用情况。第二种情况，超过监控阈值，输出当前系统的内存使用情况和故障原因，并发送告警邮件通知。

第四步，输出占用内存最高的五个进程。

第五步，将检测结果记录到日志文件中。

检测内存利用率模块流程图如图 4 所示。

该功能核心代码如下：

```
total=`free -m | grep Mem | awk '{print $2}'`
used=`free -m | grep Mem | awk '{print $3}'`
free=`free -m | grep Mem | awk '{print $4}'`
#结果小于 1.0 时前面个位数补个 0
util=`echo
"scale=2;a=($used/$total)*100;if(length(a)==scale(a)) print 0;print
a"|bc`
#监控阈值
max=50.00
```

```
#bc 对小数比较  true:1 false:0
bc=`echo "$util>$max" |bc`
```

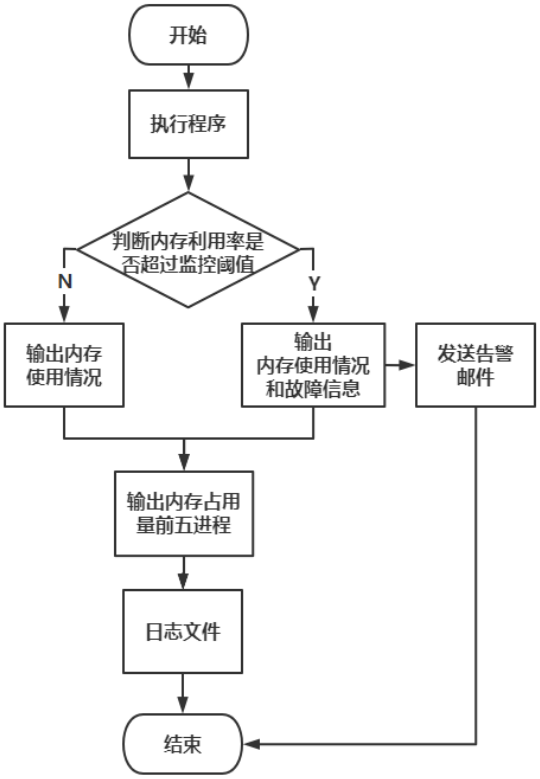

图 4　检测内存利用率流程图

（3） 检测系统服务运行状态模块

检测系统服务运行状态模块，实现步骤分为以下五步。

第一步，程序中定义需要监控的服务名称。

第二步，通过“systemctl status 服务名”命令检测服务运行状态，

第三步，做第一次判断，判断服务是否处于运行状态。第一种情况，服务正在运行，输出检测结果。第二种情况，服务未运行，尝试进行自动重启操作。

第四步，做第二次判断，判断服务是否自动重启成功。第一种情况，服务重启成功，输出重启成功信息。第二种情况，服务重启失败，输出重启失败信息，发送告警邮件通知。

第五步，将检测结果记录到日志文件中。

检测系统服务运行状态模块流程图如图 5 所示。

该功能核心代码如下：

```
    array=(docker sshd php-fpm nginx mariadb zabbix-server zabbix-agent zabbix-agent2 )
    for i in ${array[*]}
    do
        systemctl status $i &>/dev/null
        line=`echo $?`
          # 返回 0 表示服务正在运行  3 表示服务未运行
     if [[ "$line" == "0" ]];then
    echo -e "$i----服务正在运行..." |tee -a $logfile
     elif [[ "$line" == "3" ]];then
       echo -e "$i----服务未运行,正在尝试重新启动..." |tee -a $logfile
        systemctl restart $i &>/dev/null
        # 返回 5 表示服务启动失败
        if [[ "$?" == "0" ]];then
        echo -e "$i----服务自启成功,正在运行...." |tee -a $logfile
        else
           echo -e "$i----尝试自启失败" |tee -a $logfile
           echo "$ip 服务器上的$i 服务未运行，并尝试自启失败，请检查！ " |mail -s "服务状态故障通知" $mail
        fi
     else
       echo -e "$i----服务未安装!" |tee -a $logfile
     fi
    done
```

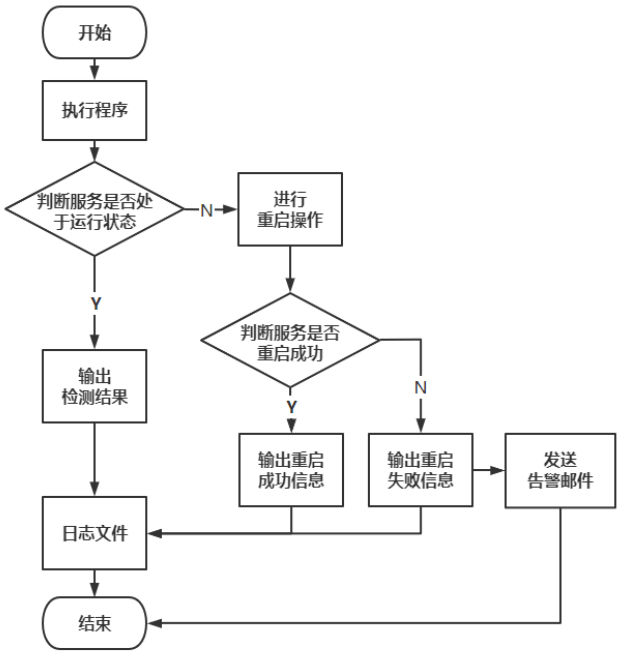

图 5　检测系统服务运行状态流程图

4.2　告警信息模块

告警信息模块中使用的 mailx 工具是 Linux 操作系统下自带的一个发送邮件的工具，本系统采用的是通过外部 QQ 邮件通知的方式，将监控状态通过邮件及时通知运维人员。实现步骤分为以下四步。

第一步，安装 mailx 工具。

第二步，修改配置文件/etc/mail.rc,添加外部邮箱参数,内容如下。

```
set from=***@qq.com （邮箱地址）
set smtp=smtp.qq.com
set smtp-auth-user=***@qq.com（邮箱地址）
set smtp-auth-password=***（邮箱独立密码）
set smtp-auth=login
```

第三步，修改/etc/hosts 文件，添加本机域名解析。

第四步，在监控程序中，设置告警邮件发送的 QQ 邮箱地址，和告警邮件内容：主机名、IP 地址、检测结果、故障信息、检测时间。

4.3 跳板机模块

跳板机模块中使用的 Webssh 工具是 Linux 操作系统下的一个开源工具，是一款具有可视化界面的远程登录服务器终端的工具，操作方便，本系统采用该工具作为跳板机，实现步骤分为以下五步。

第一步，安装 python3.6 以上的版本。

第二步，下载 Webssh 工具压缩包到本地服务器。

第三步，对工具包进行解压后执行安装程序进行安装。

第四步，防火墙开放 9999 端口（该工具的默认端口号为 9999）。

第五步，执行启动命令。

4.4 系统在线管理模块

（1） 服务管理模块

服务管理模块，主要功能是对系统服务进行操作，分别有查看服务状态、启动服务、停止服务、重启服务。

① 查看服务状态，通过“systemctl status 服务名称”命令输出该服务的运行状态。该功能核心代码如下：

```
read -p "输入服务名称[批量请用空格分隔]：" -a name
for i in ${name[*]}
do
 state=`systemctl status $i |grep -w Active |awk '{print $2}'`
 if [ "$state" = "active" ];then
        echo "$i 服务正在运行!"
    else
        echo "$i 服务未运行!"
    fi
done
```

② 启动服务，通过“systemctl start 服务名称”命令启动该服务。该功能核心代码如下：

```
read -p "输入服务名称[批量请用空格分隔]：" -a name
for i in ${name[*]}
do
 systemctl start $i &>/dev/null
    if [ $? -eq 0 ];then
        echo "$i 服务启动成功!"
    else
     echo "$i 服务启动失败!"
    fi
done
```

③ 停止服务，通过“systemctl stop 服务名称”命令停止该服务。该功能核心代码如下：

```
read -p "输入服务名称[批量请用空格分隔]：" -a name
for i in ${name[*]}
do
 systemctl stop $i &>/dev/null
 if [ $? -eq 0 ];then
      echo "$i 服务停止成功!"
    else
        echo "$i 服务停止失败!"
    fi
done
```

④ 重启服务，通过“systemctl restart 服务名称”命令重启该服务。该功能核心代码如下：

```
read -p "输入服务名称[批量请用空格分隔]：" -a name
for i in ${name[*]}
do
    systemctl restart $i &>/dev/null
    if [ $? -eq 0 ];then
        echo "$i 服务重启成功!"
    else
        echo "$i 服务重启失败!"
    fi
done
```

（2） 防火墙管理模块

防火墙管理模块，主要功能是对端口号进行操作，分别有开放端口、关闭端口、查看已开放的端口列表。

① 开放端口，首先输入端口号，其次判断该端口号是否已经开放。若查询到该端口号已开放，则输出信息，操作结束；若查询到该端口号未开放，则进行端口开放操作，输出信息，操作结束。

该功能核心代码如下：

```
read -p "输入要开放的端口号(批量请用空格分隔)：" port
array=("$port")
for i in ${array[*]}
do
 firewall-cmd --list-port |grep -w "$i" &>/dev/null
 if [ $? -eq 1 ];then
        firewall-cmd --zone=public --add-port=$i/tcp --permanent &>/dev/null
        firewall-cmd --reload &>/dev/null
        echo "$i 端口成功开放!"
    else
        echo "$i 端口已开放,无需再次开放!"
    fi
done
echo -e "\n------已开放的端口列表------"
firewall-cmd --list-port
```

② 关闭端口，首先输入端口号，其次判断该端口是否已经关闭。若查询到该端口号已关闭，则输出信息，操作结束；若查询到该端口未关闭，则进行端口关闭操作，输出信息，操作结束。

该功能核心代码如下：

```
read -p "输入要关闭的端口号(批量请用空格分隔)：" port
array=("$port")
for i in ${array[*]}
do
    firewall-cmd --list-port |grep -w "$i" &>/dev/null
    if [ $? -eq 0 ];then
        firewall-cmd --zone=public --remove-port=$i/tcp --permanent &>/dev/null
```

```
            firewall-cmd --reload &>/dev/null
            echo "$i 端口成功关闭!"
        else
            echo "$i 端口未开放!"
        fi
    done
```

5 结束语

本系统运用了 Ansible 自动化运维工具作为主要技术工具，Ansible Tower 作为 web 管理平台，用 Shell Script 脚本实现自动化检测监控模块和在线管理模块，结合功能脚本实现了一套灵活便捷的自动化运维管理系统，能及时的对系统资源进行自动化检测和监控，当发生故障问题时，进行邮件告警通知，告知运维人员，对发生故障的服务器在线进行远程维护修复，在一定程度上提高运维工作的效率，可以帮助企业更好的管理服务器。

参 考 文 献

[1] 沈晓慧,廖永行,童惠祺.自动化运维管理系统的设计与实现[J].通讯世界,2020,27(06):134-135.

[2] 庄永劼. IT 自动化运维平台设计与实现[D].电子科技大学,2020.

[3] 刘芳莉.针对业务系统及设备的运维服务平台的设计与实现[D].大连理工大学,2016.

[4] 雷竹松. 国投集团运维管理系统的设计与实现[D].大连理工大学,2019.

[5] Wang Xiaoshuang, Su Yaofeng, Li Qiang, Han Feng. Research on intelligent operation and maintenance management method of enterprise IT[J]. Journal of Physics: Conference Series,2021,1732(1).

[6] 李沁蔓.基于 Ansible 的服务器自动化运维技术研究与实现[J].电子设计工程,2020,28(13):23-26+31.

[7] 张帆. 基于 SSH 协议的字符终端会话运维审计的研究与实现[D].华北电力大学(北京),2016.

[8] 陶冲. 云端数字电路实验系统设计与实现[D].宁波大学,2019.

[9] 张超,胡玲芳.基于 SSH2-RSA 密钥验证的 SFTP 服务器配置和应用[J].电脑知识与技术,2011,7(20):4862-4863.

[10] 杨磊. 基于 Linux 系统的监控工具与其自动化测试的设计与实现[D].电子科技大学,2012.