Linux内核与Shell脚本

1. 内核简介

> 内核：在计算机科学中是一个用来管理软件发出的数据I/O（输入与输出）要求的计算机程序，将这些要求转译为数据处理的指令并交由中央处理器（CPU）及计算机中其他电子组件进行处理，是现代操作系统中最基本的部分。它是为众多应用程序提供对计算机硬件的安全访问的一部分软件，这种访问是有限的，并由内核决定一个程序在什么时候对某部分硬件操作多长时间。直接对硬件操作是非常复杂的。所以内核通常提供一种硬件抽象的方法，来完成这些操作。通过进程间通信机制及系统调用，应用进程可间接控制所需的硬件资源（特别是处理器及IO设备）。

2. GNU/Linux操作系统的基本体系结构

2.1 用户空间

最上面是用户（或应用程序）空间。

> 这是用户应用程序执行的地方。用户空间之下是内核空间，Linux 内核正是位于这里。GNU C Library（glibc）也在这里。它提供了连接内核的系统调用接口，还提供了在用户空间应用程序和内核之间进行转换的机制。这点非常重要，因为内核和用户空间的应用程序使用的是不同的保护地址空间。每个用户空间的进程都使用自己的虚拟地址空间，而内核则占用单独的地址空间。

2.2 Linux内核的体系结构

内核是操作系统的核心，具有很多最基本功能，它负责管理系统的进程、内存、设备驱动程序、文件和网络系统，决定着系统的性能和稳定性。

Linux 内核由如下几部分组成：内存管理、进程管理、设备驱动程序、文件系统和网络管理等。

2.2.1系统调用接口（System Call Interface 简称 SCI)

SCI 层提供了某些机制执行从用户空间到内核的函数调用。这个接口依赖于体系结构，甚至在相同的处理器家族内也是如此。

SCI 实际上是一个非常有用的函数调用多路复用和多路分解服务。

在 ./linux/kernel 中您可以找到 SCI 的实现，并在 ./linux/arch 中找到依赖于体系结构的部分。

2.2.2 文件管理

对任何一台计算机而言，其内存以及其它资源都是有限的。为了让有限的物理内存满足应用程序对内存的大需求量，Linux 采用了称为“虚拟内存”的内存管理方式。Linux 将内存划分为容易处理的“内存页”（对于大部分体系结构来说都是 4KB）。Linux 包括了管理可用内存的方式，以及物理和虚拟映射所使用的硬件机制。不过内存管理要管理的可不止 4KB 缓冲区。Linux 提供了对 4KB 缓冲区的抽象

> 例如 slab 分配器。这种内存管理模式使用 4KB 缓冲区为基数，然后从中分配结构，并跟踪内存页使用情况，比如哪些内存页是满的，哪些页面没有完全使用，哪些页面为空。这样就允许该模式根据系统需要来动态调整内存使用。

为了支持多个用户使用内存，有时会出现可用内存被消耗光的情况。由于这个原因，页面可以移出内存并放入磁盘中。这个过程称为交换，因为页面会被从内存交换到硬盘上。内存管理的源代码可以在./linux/mm 中找到。

2.2.3. 进程管理

进程实际是某特定应用程序的一个运行实体。在 Linux 系统中，能够同时运行多个进程，Linux 通过在短的时间间隔内轮流运行这些进程而实现"多任务"。这一短的时间间隔称为"时间片"，让进程轮流运行的方法称为"进程调度"，完成调度的程序称为调度程序。

> 进程调度控制进程对CPU的访问。当需要选择下一个进程运行时，由调度程序选择最值得运行的进程。可运行进程实际上是仅等待CPU资源的进程，如果某个进程在等待其它资源，则该进程是不可运行进程。Linux使用了比较简单的基于优先级的进程调度算法选择新的进程。
>
> 通过多任务机制，每个进程可认为只有自己独占计算机，从而简化程序的编写。每个进程有自己单独的地址空间，并且只能由这一进程访问，这样，操作系统避免了进程之间的互相干扰以及"坏"程序对系统可能造成的危害。 为了完成某特定任务，有时需要综合两个程序的功能，例如一个程序输出文本，而另一个程序对文本进行排序。为此，操作系统还提供进程间的通讯机制来帮助完成这样的任务。Linux 中常见的进程间通讯机制有信号、管道、共享内存、信号量和套接字等。
>
> 内核通过 SCI 提供了一个应用程序编程接口（API）来创建一个新进程（fork、exec 或 Portable Operating System Interface [POSIX] 函数），停止进程（kill、exit），并在它们之间进行通信和同步（signal 或者 POSIX 机制）。

2.2.4 文件系统

和 DOS 等操作系统不同，Linux 操作系统中单独的文件系统并不是由驱动器号或驱动器名称（如 A: 或 C: 等）来标识的。相反，和 UNIX 操作系统一样，Linux 操作系统将独立的文件系统组合成了一个层次化的树形结构，并且由一个单独的实体代表这一文件系统。

Linux 将新的文件系统通过一个称为"挂装"或"挂上"的操作将其挂装到某个目录上，从而让不同的文件系统结合成为一个整体。Linux 操作系统的一个重要特点是它支持许多不同类型的文件系统。

Linux 中最普遍使用的文件系统是 Ext2，它也是 Linux 土生土长的文件系统。但 Linux 也能够支持 FAT、VFAT、FAT32、MINIX 等不同类型的文件系统，从而可以方便地和其它操作系统交换数据。由于 Linux 支持许多不同的文件系统，并且将它们组织成了一个统一的虚拟文件系统.

虚拟化文件系统：

> 虚拟文件系统（VFS）是 Linux 内核中非常有用的一个方面，因为它为文件系统提供了一个通用的接口抽象。VFS 在 SCI 和内核所支持的文件系统之间提供了一个交换层。即VFS 在用户和文件系统之间提供了一个交换层。
>
> 虚拟化文件系统隐藏了各种硬件的具体细节，把文件系统操作和不同文件系统的具体实现细节分离了开来，为所有的设备提供了统一的接口，VFS提供了多达数十种不同的文件系统。虚拟文件系统可以分为逻辑文件系统和设备驱动程序。逻辑文件系统指Linux所支持的文件系统，如ext2,fat等，设备驱动程序指为每一种硬件控制器所编写的设备驱动程序模块。

2.2.5 设备驱动程序

设备驱动程序是 Linux 内核的主要部分。和操作系统的其它部分类似，设备驱动程序运行在高特权级的处理器环境中，从而可以直接对硬件进行操作，但正因为如此，任何一个设备驱动程序的错误都可能导致操作系统的崩溃。设备驱动程序实际控制操作系统和硬件设备之间的交互。设备驱动程序提供一组操作系统可理解的抽象接口完成和操作系统之间的交互，而与硬件相关的具体操作细节由设备驱动程序完成。一般而言，设备驱动程序和设备

的控制芯片有关，例如，如果计算机硬盘是 SCSI 硬盘，则需要使用 SCSI 驱动程序，而不是 IDE 驱动程序。

2.2.6 网络接口程序

提供了对各种网络标准的存取和各种网络硬件的支持。网络接口可分为网络协议和网络驱动程序。网络协议部分负责实现每一种可能的网络传输协议。众所周知，TCP/IP 协议是 Internet 的标准协议，同时也是事实上的工业标准。Linux 的网络实现支持 BSD 套接字，支持全部的TCP/IP协议。Linux内核的网络部分由BSD套接字、网络协议层和网络设备驱动程序组成。

网络设备驱动程序负责与硬件设备通讯，每一种可能的硬件设备都有相应的设备驱动程序。

3. shell脚本

3.1 简介与运行环境

Shell脚本是一种为shell编写的脚本程序。通常所说的shell都是指shell脚本，但是shell与shell script是两个不同的概念

Shell编程跟JavaScript，php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。

Linux的shell种类众多，常见的有：

- Bourne Shell（/usr/bin/sh或/bin/sh）
- Bourne Again Shell（/bin/bash）
- C Shell（/usr/bin/csh）
- K Shell（/usr/bin/ksh）
- Shell for Root（/sbin/sh）
- ……

Bash，也就是 Bourne Again Shell，由于易用和免费，Bash 在日常工作中被广泛使用。同时，Bash 也是大多数Linux 系统默认的 Shell。

在一般情况下，人们并不区分 Bourne Shell 和 Bourne Again Shell，所以，像 **#!/bin/sh**，它同样也可以改为 **#!/bin/bash**。

#! 告诉系统其后路径所指定的程序即是解释此脚本文件的 Shell 程序。

3.2 Shell脚本的使用

```
#    !/bin/bash ##声明 bash 脚本
##demo ##注释
echo "Hello World !"  ##输出
name="fame老师"   ## 定义变量
echo $myUrl  ##    输出变量
echo "I am ${name}'s friend" ##字符串拼接
echo ""  ## 换行

names=("Zero老师" "Shine York老师" "Pack老师") ##定义数组
echo ${names[@]} ##遍历数
echo "I am ${names[1]}'s friend" ##第二个元素
echo "I have ${#names[@]} friends" ##数组长度
echo ""

for var in ${names[@]}; ##循环数组
do
    if test $var = 'Peter' ##字符串相等
```

```
    then
        echo "I am Peter"
    else
        echo "I am ${var}'s friend"
    fi
done
echo ""
```

重定向：

```
1、test 'aa' -eq "bb" > out ##命令输出到 out 文件，报错信息并不会进入 out
2、test 'aa' -eq "bb" > out 2>&1 ##将 stderr 合并到 stdout，则报错信息进入了 out
```

3.3 mysql定时备份

需求：

```
1、可以自定义每天什么时间来执行指令，进行备份数据库wjy到/home目录下（这个目录可以根据具体情况来自定义）；
2、备份开始和备份结束能够给出相对应的提示信息；
3、备份后的文件要求以备份时间为文件名，并打包成.tar.gz 的形式，比如：2019-02-21.tar.gz
4、在备份的同时，检查是否有大于10天前备份的数据库文件，如果有就将其删除；
```

脚本代码：

```
echo "===开始备份==="
BACKUP=/home
DATETIME=$(date +%Y-%m-%d)
echo "===备份的路径是：$BACKUP/$DATETIME/$DATETIME.tar.gz"

#主机
HOST=127.0.0.1
#用户名
DB_USER=root
#密码
DB_PWD=root
#备份数据库名
DATABASE=test
#创建备份的路径
#如果备份的路径文件夹存在，就使用，否则就创建
[ ! -d "$BACKUP/$DATETIME" ] && mkdir -p "$BACKUP/$DATETIME"
#执行mysql的备份数据库的指令
mysqldump -u${DB_USER} -p${DB_PWD} --host=$HOST $DATABASE | gzip >
$BACKUP/$DATETIME/$DATETIME.sql.gz
#打包备份文件
cd $BACKUP
tar -zcvf $DATETIME.tar.gz $DATETIME
#删除临时目录
rm -rf $BACKUP/$DATETIME

#删除10天前的备份文件
find $BACKUP -mtime +10 -name "*.tar.gz" -exec rm -rf {} \;
echo "===备份文件成功==="
```

设置每天4点备份mysql数据

```
#创建定时任务
crontab -e
0 4 * * * /data/dbdata/backup_mysql.sh
```