面试题一

1、什么是Linux?

Linux是一套免费使用和自由传播的类Unix操作系统，是一个基于POSIX和Unix的多用户、多任务、支持多线程和多CPU的操作系统。它能运行主要的Unix工具软件、应用程序和网络协议。它支持32位和64位硬件。Linux继承了Unix以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

2、Unix和Linux有什么区别?

Linux和Unix都是功能强大的操作系统，都是应用广泛的服务器操作系统，有很多相似之处，甚至有一部分人错误地认为Unix和Linux操作系统是一样的，然而，事实并非如此，以下是两者的区别。

1、开源性

Linux是一款开源操作系统，不需要付费，即可使用；Unix是一款对源码实行知识产权保护的传统商业软件，使用需要付费授权使用。

2、跨平台性

Linux操作系统具有良好的跨平台性能，可运行在多种硬件平台上；Unix操作系统跨平台性能较弱，大多需与硬件配套使用。

3、可视化界面

Linux除了进行命令行操作，还有窗体管理系统；Unix只是命令行下的系统。

4、硬件环境

Linux操作系统对硬件的要求较低，安装方法更易掌握；Unix对硬件要求比较苛刻，安装难度较大。

用户群体

Linux的用户群体很广泛，个人和企业均可使用；Unix的用户群体比较窄，多是安全性要求高的大型企业使用，如银行、电信部门等，或者Unix硬件厂商使用，如Sun等。

相比于Unix操作系统，Linux操作系统更受广大计算机爱好者的喜爱，主要原因是Linux操作系统具有Unix操作系统的全部功能，并且能够在普通PC计算机上实现全部的Unix特性，开源免费的特性，更容易普及使用!

3、什么是 Linux 内核?

Linux 系统的核心是内核。内核控制着计算机系统上的所有硬件和软件，在必要时分配硬件，并根据需要执行软件。

1、系统内存管理

2、应用程序管理

3、硬件设备管理

4、文件系统管理

4、Linux的基本组件是什么?

就像任何其他典型的操作系统一样，Linux拥有所有这些组件：内核，shell和GUI，系统实用程序和应用程序。Linux比其他操作系统更具优势的是每个方面都附带其他功能，所有代码都可以免费下载。

5、Linux 的体系结构?

从大的方面讲，Linux 体系结构可以分为两块：

- 用户空间(User Space)： 用户空间又包括用户的应用程序(User Applications)、C 库(C Library) 。
- 内核空间(Kernel Space)： 内核空间又包括系统调用接口(System Call Interface)、内核(Kernel)、平台架构相关的代码(Architecture-Dependent Kernel Code)

6、BASH和DOS之间的基本区别是什么?

BASH和DOS控制台之间的主要区别在于3个方面：

- BASH命令区分大小写，而DOS命令则不区分;
- 在BASH下，/ character是目录分隔符，\作为转义字符。在DOS下，/用作命令参数分隔符，\是目录分隔符
- DOS遵循命名文件中的约定，即8个字符的文件名后跟一个点，扩展名为3个字符。BASH没有遵循这样的惯例。

7、Linux 开机启动过程?

> 了解即可。

1、主机加电自检，加载 BIOS 硬件信息。

2、读取 MBR 的引导文件(GRUB、LILO)。

3、引导 Linux 内核。

4、运行第一个进程 init (进程号永远为 1)。

5、进入相应的运行级别。

6、运行终端，输入用户名和密码。

8、Linux系统缺省的运行级别?

- 关机。
- 单机用户模式。
- 字符界面的多用户模式(不支持网络)。
- 字符界面的多用户模式。
- 未分配使用。
- 图形界面的多用户模式。
- 重启。

9、Linux 使用的进程间通信方式?

- 1、管道(pipe)、流管道(s_pipe)、有名管道(FIFO)。
- 2、信号(signal) 。
- 3、消息队列。
- 4、共享内存。
- 5、信号量。
- 6、套接字(socket) 。

10、Linux 有哪些系统日志文件?

比较重要的是 `/var/log/messages` 日志文件。

> 该日志文件是许多进程日志文件的汇总，从该文件可以看出任何入侵企图或成功的入侵。
> 另外，如果胖友的系统里有 ELK 日志集中收集，它也会被收集进去。

11、Linux系统安装多个桌面环境有帮助吗?

通常，一个桌面环境，如KDE或Gnome，足以在没有问题的情况下运行。尽管系统允许从一个环境切换到另一个环境，但这对用户来说都是优先考虑的问题。有些程序在一个

环境中工作而在另一个环境中无法工作，因此它也可以被视为选择使用哪个环境的一个因素。

12、什么是交换空间?

交换空间是Linux使用的一定空间，用于临时保存一些并发运行的程序。当RAM没有足够的内存来容纳正在执行的所有程序时，就会发生这种情况。

13、什么是root帐户?

root帐户就像一个系统管理员帐户，允许你完全控制系统。你可以在此处创建和维护用户帐户，为每个帐户分配不同的权限。每次安装Linux时都是默认帐户。

14、什么是LILO?

LILO是Linux的引导加载程序。它主要用于将Linux操作系统加载到主内存中，以便它可以开始运行。

15、什么是BASH?

BASH是Bourne Again SHell的缩写。它由Steve Bourne编写，作为原始Bourne Shell（由/ bin / sh表示）的替代品。它结合了原始版本的Bourne Shell的所有功能，以及其他功能，使其更容易使用。从那以后，它已被改编为运行Linux的大多数系统的默认shell。

16、什么是CLI?

命令行界面（英语：**command-line interface**，缩写]：CLI）是在图形用户界面得到普及之前使用最为广泛的用户界面，它通常不支持鼠标，用户通过键盘输入指令，计算机接收到指令后，予以执行。也有人称之为字符用户界面（CUI）。

通常认为，命令行界面（CLI）没有图形用户界面（GUI）那么方便用户操作。因为，命令行界面的软件通常需要用户记忆操作的命令，但是，由于其本身的特点，命令行界面要较图形用户界面节约计算机系统的资源。在熟记命令的前提下，使用命令行界面往往要较使用图形用户界面的操作速度要快。所以，图形用户界面的操作系统中，都保留着可选的命令行界面。

17、什么是GUI?

图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口）是指采用图形方式显示的计算机操作用户界面。

图形用户界面是一种人与计算机通信的界面显示格式，允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，以选择命令、调用文件、启动程序或执行其它一些日常任务。与通过键盘输入文本或字符命令来完成例行任务的字符界面相比，图形用户界面有许多优点。

18、开源的优势是什么?

开源允许你将软件（包括源代码）免费分发给任何感兴趣的人。然后，人们可以添加功能，甚至可以调试和更正源代码中的错误。它们甚至可以让它运行得更好，然后再次自由地重新分配这些增强的源代码。这最终使社区中的每个人受益。

19、简单 Linux 文件系统?

在 Linux 操作系统中，所有被操作系统管理的资源，例如网络接口卡、磁盘驱动器、打印机、输入输出设备、普通文件或是目录都被看作是一个文件。

也就是说在 Linux 系统中有一个重要的概念：一切都是文件。其实这是 Unix 哲学的一个体现，而 Linux 是重写 Unix 而来，所以这个概念也就传承了下来。在 Unix 系统中，把一切资源都看作是文件，包括硬件设备。UNIX系统把每个硬件都看成是一个文件，通常称为设备文件，这样用户就可以用读写文件的方式实现对硬件的访问。

Linux 支持 5 种文件类型，如下图所示：

20、Linux 的目录结构是怎样的?

> 这个问题，一般不会问。更多是实际使用时，需要知道。

Linux 文件系统的结构层次鲜明，就像一棵倒立的树，最顶层是其根目录：

常见目录说明：

- /bin： 存放二进制可执行文件(ls,cat,mkdir等)，常用命令一般都在这里；
- /etc： 存放系统管理和配置文件；
- /home： 存放所有用户文件的根目录，是用户主目录的基点，比如用户user的主目录就是/home/user，可以用~user表示；
- /usr： 用于存放系统应用程序；
- /opt： 额外安装的可选应用程序包所放置的位置。一般情况下，我们可以把tomcat等都安装到这里；
- /proc： 虚拟文件系统目录，是系统内存的映射。可直接访问这个目录来获取系统信息；
- /root： 超级用户（系统管理员）的主目录（特权阶级o）；
- /sbin: 存放二进制可执行文件，只有root才能访问。这里存放的是系统管理员使用的系统级别的管理命令和程序。如ifconfig等；
- /dev： 用于存放设备文件；

- /mnt： 系统管理员安装临时文件系统的安装点，系统提供这个目录是让用户临时挂载其他的文件系统；
- /boot： 存放用于系统引导时使用的各种文件；
- /lib： 存放着和系统运行相关的库文件 ；
- /tmp： 用于存放各种临时文件，是公用的临时文件存储点；
- /var： 用于存放运行时需要改变数据的文件，也是某些大文件的溢出区，比方说各种服务的日志文件（系统启动日志等。）等；
- /lost+found： 这个目录平时是空的，系统非正常关机而留下“无家可归”的文件（windows下叫什么.chk）就在这里。

21、什么是 inode？

> 一般来说，面试不会问 inode 。但是 inode 是一个重要概念，是理解 Unix/Linux 文件系统和硬盘储存的基础。

理解inode，要从文件储存说起。

文件储存在硬盘上，硬盘的最小存储单位叫做"扇区"（Sector）。每个扇区储存512字节（相当于0.5KB）。

操作系统读取硬盘的时候，不会一个个扇区地读取，这样效率太低，而是一次性连续读取多个扇区，即一次性读取一个"块"（block）。这种由多个扇区组成的"块"，是文件存取的最小单位。"块"的大小，最常见的是4KB，即连续八个 sector组成一个 block。

文件数据都储存在"块"中，那么很显然，我们还必须找到一个地方储存文件的元信息，比如文件的创建者、文件的创建日期、文件的大小等等。这种储存文件元信息的区域就叫做inode，中文译名为"索引节点"。

每一个文件都有对应的inode，里面包含了与该文件有关的一些信息。

简述 Linux 文件系统通过 i 节点把文件的逻辑结构和物理结构转换的工作过程?

> 如果看的一脸懵逼，也没关系。一般来说，面试官不太会问这个题目。

Linux 通过 inode 节点表将文件的逻辑结构和物理结构进行转换。

- inode 节点是一个 64 字节长的表，表中包含了文件的相关信息，其中有文件的大小、文件所有者、文件的存取许可方式以及文件的类型等重要信息。在 inode 节点表中最重要的内容是磁盘地址表。在磁盘地址表中有 13 个块号，文件将以块号在磁盘地址表中出现的顺序依次读取相应的块。
- Linux 文件系统通过把 inode 节点和文件名进行连接，当需要读取该文件时，文件系统在当前目录表中查找该文件名对应的项，由此得到该文件相对应的 inode 节点

号，通过该 inode 节点的磁盘地址表把分散存放的文件物理块连接成文件的逻辑结构。

22、什么是硬链接和软链接?

1）硬链接

由于 Linux 下的文件是通过索引节点(inode)来识别文件，硬链接可以认为是一个指针，指向文件索引节点的指针，系统并不为它重新分配 inode 。每添加一个一个硬链接，文件的链接数就加 1 。

- 不足：1）不可以在不同文件系统的文件间建立链接；2）只有超级用户才可以为目录创建硬链接。

2）软链接

软链接克服了硬链接的不足，没有任何文件系统的限制，任何用户可以创建指向目录的符号链接。因而现在更为广泛使用，它具有更大的灵活性，甚至可以跨越不同机器、不同网络对文件进行链接。

- 不足：因为链接文件包含有原文件的路径信息，所以当原文件从一个目录下移到其他目录中，再访问链接文件，系统就找不到了，而硬链接就没有这个缺陷，你想怎么移就怎么移；还有它要系统分配额外的空间用于建立新的索引节点和保存原文件的路径。

实际场景下，基本是使用软链接。总结区别如下：

- 硬链接不可以跨分区，软件链可以跨分区。
- 硬链接指向一个 inode 节点，而软链接则是创建一个新的 inode 节点。
- 删除硬链接文件，不会删除原文件，删除软链接文件，会把原文件删除。

23、RAID 是什么?

RAID 全称为独立磁盘冗余阵列(Redundant Array of Independent Disks)，基本思想就是把多个相对便宜的硬盘组合起来，成为一个硬盘阵列组，使性能达到甚至超过一个价格昂贵、 容量巨大的硬盘。RAID 通常被用在服务器电脑上，使用完全相同的硬盘组成一个逻辑扇区，因此操作系统只会把它当做一个硬盘。

RAID 分为不同的等级，各个不同的等级均在数据可靠性及读写性能上做了不同的权衡。在实际应用中，可以依据自己的实际需求选择不同的 RAID 方案。

24、一台 Linux 系统初始化环境后需要做一些什么安全工作?

- 1、添加普通用户登陆，禁止 root 用户登陆，更改 SSH 端口号。

> 修改 SSH 端口不一定绝对哈。当然，如果要暴露在外网，建议改下。

- 2、服务器使用密钥登陆，禁止密码登陆。
- 3、开启防火墙，关闭 SElinux ，根据业务需求设置相应的防火墙规则。
- 4、装 fail2ban 这种防止 SSH 暴力破击的软件。
- 5、设置只允许公司办公网出口 IP 能登陆服务器(看公司实际需要)

> 也可以安装 VPN 等软件，只允许连接 VPN 到服务器上。

- 6、修改历史命令记录的条数为 10 条。
- 7、只允许有需要的服务器可以访问外网，其它全部禁止。
- 8、做好软件层面的防护。
  - 8.1 设置 nginx_waf 模块防止 SQL 注入。
  - 8.2 把 Web 服务使用 www 用户启动，更改网站目录的所有者和所属组为 www 。

25、什么叫 CC 攻击？什么叫 DDOS 攻击?

CC 攻击，主要是用来攻击页面的，模拟多个用户不停的对你的页面进行访问，从而使你的系统资源消耗殆尽。

DDOS 攻击，中文名叫分布式拒绝服务攻击，指借助服务器技术将多个计算机联合起来作为攻击平台，来对一个或多个目标发动 DDOS 攻击。

> 攻击，即是通过大量合法的请求占用大量网络资源，以达到瘫痪网络的目的。

怎么预防 CC 攻击和 DDOS 攻击?

防 CC、DDOS 攻击，这些只能是用硬件防火墙做流量清洗，将攻击流量引入黑洞。

> 流量清洗这一块，主要是买 ISP 服务商的防攻击的服务就可以，机房一般有空余流量，我们一般是买服务，毕竟攻击不会是持续长时间。

26、什么是网站数据库注入?

由于程序员的水平及经验参差不齐，大部分程序员在编写代码的时候，没有对用户输入数据的合法性进行判断。

应用程序存在安全隐患。用户可以提交一段数据库查询代码，根据程序返回的结果，获得某些他想得知的数据，这就是所谓的 SQL 注入。

SQL注入，是从正常的 WWW 端口访问，而且表面看起来跟一般的 Web 页面访问没什么区别，如果管理员没查看日志的习惯，可能被入侵很长时间都不会发觉。

如何过滤与预防?

数据库网页端注入这种，可以考虑使用 nginx_waf 做过滤与预防。

27、Shell 脚本是什么?

一个 Shell 脚本是一个文本文件，包含一个或多个命令。作为系统管理员，我们经常需要使用多个命令来完成一项任务，我们可以添加这些所有命令在一个文本文件(Shell 脚本)来完成这些日常工作任务。

28、可以在 Shell 脚本中使用哪些类型的变量?

在 Shell 脚本，我们可以使用两种类型的变量：

系统定义变量

> 系统变量是由系统系统自己创建的。这些变量通常由大写字母组成，可以通过 set 命令查看。

用户定义变量

> 用户变量由系统用户来生成和定义，变量的值可以通过命令 "echo $<变量名>" 查看。

29、Shell 脚本中 if 语法如何嵌套?

```
if [ 条件 ]
then
命令1
命令2
……
else
if [ 条件 ]
then
命令1
命令2
…….
else
命令1
命令2
……
fi
fi
```

30、Shell 脚本中 case 语句的语法?

```
case 变量 in
值1)
命令1
命令2
……
最后命令
!!
值2)
命令1
命令2
……
最后命令
;;
esac
```

31、Shell 脚本中 for 循环语法?

```
for 变量 in 循环列表
do
命令1
命令2
…….
最后命令
done
```

32、Shell 脚本中 while 循环语法?

如同 for 循环，while 循环只要条件成立就重复它的命令块。

不同于 for循环，while 循环会不断迭代，直到它的条件不为真。

基础语法:

```
while [ 条件 ]
do
命令…
done
```

33、如何使脚本可执行?

使用 chmod 命令来使脚本可执行。例子如下：`chmod a+x myscript.sh`

34、在 Shell 脚本如何定义函数呢?

函数是拥有名字的代码块。当我们定义代码块，我们就可以在我们的脚本调用函数名字，该块就会被执行。示例如下所示：

```
$ diskusage () { df -h ; }
译注：下面是我给的shell函数语法，原文没有
[ function ] 函数名 [()]
{
命令;
[return int;]
}
```

35、判断一文件是不是字符设备文件，如果是将其拷贝到 /dev 目录下?

```
#!/bin/bash
read -p "Input file name: " FILENAME
if [ -c "$FILENAME" ];then
    cp $FILENAME /dev
fi
```

36、添加一个新组为 class1，然后添加属于这个组的 30 个用户，用户名的形式为 stdxx，其中 xx 从 01 到 30？

```
#!/bin/bash
groupadd class1
for((i=1;i<31;i++))
do
        if [ $i -le 10 ];then
                useradd -g class1 std0$i
        else
                useradd -g class1 std$i
        fi
done
```

37、写一个 sed 命令，修改 /tmp/input.txt 文件的内容?

要求：

- 删除所有空行。
- 一行中，如果包含 “11111”，则在 “11111” 前面插入 “AAA”，在 “11111” 后面插入 “BBB” 。比如：将内容为 0000111112222 的一行改为 0000AAA11111BBB2222 。

```
[root@~]## cat -n /tmp/input.txt
     1  000011111222
     2
     3  000011111222222
     4  11111000000222
     5
     6
     7  11111111111122222222222
     8  2211111111
     9  112222222
    10  1122
    11

## 删除所有空行命令
[root@~]## sed '/^$/d' /tmp/input.txt
000011111222
000011111222222
11111000000222
11111111111122222222222
2211111111
112222222
1122

## 插入指定的字符
[root@~]## sed 's#\(11111\)#AAA\1BBB#g' /tmp/input.txt
0000AAA11111BBB222
0000AAA11111BBB222222
AAA11111BBB000000222
AAA11111BBBAAA11111BBB11122222222222
22AAA11111BBB111
112222222
1122
```

38、用户进程间通信主要哪几种方式?

(1)管道(Pipe):管道可用于具有亲缘关系进程间的通信,允许一个进程和另一个与它有共同祖 先的进程之间进行通信。

(2)命名管道(named pipe):命名管道克服了管道没有名字的限制,因此,除具有管道所具有的 功能外,它还允许无亲缘关系进程间的通信。命名管道在文件系统中有对应的文件名。命名 管道通过命令 mkfifo 或系统调用 mkfifo 来创建。

(3)信号(Signal):信号是比较复杂的通信方式,用于通知接受进程有某种事件发生,除了用于进 程间通信外,进程还可以发送信号给进程本身;linux 除了支持 Unix 早期信号语义函数 sigal 外,还支持语义符合 Posix.1 标准的信号函数 sigaction(实际上,该函数是基于 BSD 的,BSD 为了实现可靠信号机制,又能够统一对外接口,用 sigaction 函数重新实现了 signal 函数)。

(4)消息(Message)队列:消息队列是消息的链接表,包括 Posix 消息队列 system V 消息队列。有足够权限的进程可以向队列中添加消息,被赋予读权限的进程则可以读走队列中的消 息。消息队列克服了信号承载信息量少,管道只能承载无格式字节流以及缓冲区大小受限等 缺

(5)共享内存:使得多个进程可以访问同一块内存空间,是最快的可用 IPC 形式。是针对其他 通信机制运行效率较低而设计的。往往与其它通信机制,如信号量结合使用,来达到进程间 的同步及互斥。

(6)信号量(semaphore):主要作为进程间以及同一进程不同线程之间的同步手段。

(7)套接字(Socket):更为一般的进程间通信机制,可用于不同机器之间的进程间通信。起初是 由 Unix 系统的 BSD 分支开发出来的,但现在一般可以移植到其它类 Unix 系统上:Linux 和 System V 的变种都支持套接字。

39、通过伙伴系统申请内核内存的函数有哪些?

在物理页面管理上实现了基于区的伙伴系统(zone based buddy system)。对不同区的内存 使用单独的伙伴系统(buddy system)管理,而且独立地监控空闲页。
相应接口 `alloc_pages(gfp_mask, order)`,`_ _get_free_pages(gfp_mask, order)` 等。

40、Linux 虚拟文件系统的关键数据结构有哪些?(至少写出四个)

- struct super_block
- struct inode
- struct fil
- struct dentry

41、对文件或设备的操作函数保存在那个数据结构中?

struct file_operations

42、Linux 中的文件包括哪些?

- 执行文件
- 普通文件

- 目录文件
- 链接文件和设备文件
- 管道文件

43、创建进程的系统调用有那些?

- clone()
- fork()
- fork()

系统调用服务例程:

- sys_clone
- sys_fork
- sys_vfork

44、调用 schedule()进行进程切换的方式有几种?

1.系统调用 do_fork();

2.定时中断 do_timer();

3.唤醒进程 wake_up_process

4.改变进程的调度策略 setscheduler();

5.系统调用礼让 sys_sched_yield();

45、Linux 调度程序是根据进程的动态优先级还是静态优先级来调度进程的?

Liunx 调度程序是根据根据进程的动态优先级来调度进程的,但是动态优先级又是根据静态 优先级根据算法计算出来的,两者是两个相关联的值。因为高优先级的进程总是比低优先级

的进程先被调度,为防止多个高优先级的进程占用 CPU 资源,导致其他进程不能占有 CPU, 所以引用动态优先级概念

46、进程调度的核心数据结构是哪个?

struct runqueue

47、如何加载、卸载一个模块?

- insmod 加载

- rmmod 卸载

48、模块和应用程序分别运行在什么空间?

模块运行在内核空间,应用程序运行在用户空间

49、Linux 中的浮点运算由应用程序实现还是内核实现?

应用程序实现,Linux 中的浮点运算是利用数学库函数实现的,库函数能够被应用程序链接后 调用,不能被内核链接调用。这些运算是在应用程序中运行的,然后再把结果反馈给系统。 Linux 内核如果一定要进行浮点运算,需要在建立内核时选上 math-emu,使用软件模拟计算 浮点运算,据说这样做的代价有两个:用户在安装驱动时需要重建内核,可能会影响到其他的 应用程序,使得这些应用程序在做浮点运算的时候也使用 math-emu,大大的降低了效率。

50、模块程序能否使用可链接的库函数?

模块程序运行在内核空间,不能链接库函数。

51、TLB 中缓存的是什么内容?

TLB,页表缓存,当线性地址被第一次转换成物理地址的时候,将线性地址和物理地址的对应 放到 TLB 中,用于下次访问这个线性地址时,加快转换速度。

52、Linux 中有哪几种设备?

字符设备和块设备。网卡是例外,他不直接与设备文件对应,mknod 系统调用用来创建设备 文件。

53、字符设备驱动程序的关键数据结构是哪个?

字符设备描述符 struct cdev,cdev_alloc()用于动态的分配 cdev 描述符,cdev_add()用于注 册一个 cdev 描述符,cdev 包含一个 struct kobject 类型的数据结构它是核心的数据结构

54、设备驱动程序包括哪些功能函数?

- open()
- read()
- write()
- llseek()
- realse()

55、如何唯一标识一个设备?

Linux 使用一个设备编号来唯一的标示一个设备,设备编号分为:主设备号和次设备号,一般主 设备号标示设备对应的驱动程序,次设备号对应设备文件指向的设备,在内核中使用 dev_t 来 表示设备编号,一般它是 32 位长度,其中 12 位用于表示主设备号,20 位用于表示次设备号, 利用 MKDEV(int major,int minor);用于生成一个 dev_t 类型的对象。

56、Linux 通过什么方式实现系统调用?

靠软件中断实现的,首先,用户程序为系统调用设置参数,其中一个编号是系统调用编号,参数 设置完成后,程序执行系统调用指令,x86 上的软中断是有 int 产生的,这个指令会导致一个异 常,产生一个事件,这个事件会导致处理器跳转到内核态并跳转到一个新的地址。并开始处 理那里的异常处理程序,此时的异常处理就是系统调用程序。

57、Linux 软中断和工作队列的作用是什么?

Linux 中的软中断和工作队列是中断处理。

1.软中断一般是“可延迟函数”的总称,它不能睡眠,不能阻塞,它处于中断上下文,不能进城切 换,软中断不能被自己打断,只能被硬件中断打断(上半部),可以并发的运行在多个 CPU 上。 所以软中断必须设计成可重入的函数,因此也需要自旋锁来保护其数据结构。

2.工作队列中的函数处在进程上下文中,它可以睡眠,也能被阻塞,能够在不同的进程间切换。已完成不同的工作。 可延迟函数和工作队列都不能访问用户的进程空间,可延时函数在执行时不可能有任何正在 运行的进程,工作队列的函数有内核进程执行,他不能访问用户空间地址

58、Linux开机启动过程?

1）主机加电自检，加载BOLS硬件信息

2）读取MBR的引导文件（grub，lilo）

3）引导linux内核

4）运行第一个进程init（进程号永远为1）

5）进入相应的运行级别

6）运行终端，输入用户名和密码

59、Linux系统缺省的运行级别

0.关机

1.单机用户模式

2.字符界面的多用户模式（不支持网络）

3.字符界面的多用户模式

4.未分配使用

5.图形界面的多用户模式

6.重启

60、Linux系统是由那些部分组成?

Linux系统内核，shell，文件系统和应用程序四部分组成

61、硬链接和软链接有什么区别?

1）硬链接不可以跨分区，软件链可以跨分区

2）硬链接指向一个i节点，而软链接则是创建一个新的i节点

3）删除硬链接文件，不会删除原文件，删除软链接文件，会把原文件删除

62、如何规划一台Linux主机，步骤是怎样?

1、确定机器是做什么用的，比如是做 WEB 、DB、还是游戏服务器。

> 不同的用途，机器的配置会有所不同。

2、确定好之后，就要定系统需要怎么安装，默认安装哪些系统、分区怎么做。

3、需要优化系统的哪些参数，需要创建哪些用户等等的。

63、查看系统当前进程连接数?

```
netstat -an | grep ESTABLISHED | wc -l
```

64、如何在/usr目录下找出大小超过10MB的文件?

```
# find /usr -size +10M
```

65、添加一条到192.168.3.0/24的路由，网关为192.168.1.254?

```
route add -net 192.168.3.0/24 netmask 255.255.255.0 gw 192.168.1.254
```

66、如何在/var目录下找出90天之内未被访问过的文件?

```
find /var \! -atime -90
```

67、如何在/home目录下找出120天之前被修改过的文件?

```
find /home  -mtime +120
```

68、在整个目录树下查找文件“core”，如发现则无需提示直接删除它们。

```
find / -name core -exec rm {} \;
```

69、有一普通用户想在每周日凌晨零点零分定期备份/user/backup到/tmp目录下，该用户应如何做?

```
crontab -e
0 0 * * 7 /bin/cp /user/backup /tmp
```

70、每周一下午三点将/tmp/logs目录下面的后缀为*.log的所有文件rsync同步到备份服务器192.168.1.100中同样的目录下面，crontab配置项该如何写?

```
00 15 * * 1 rsync -avzP /tmp/logs/*.log root@192.168.1.100:/tmp/logs
```

71、找到/tmp/目录下面的所有名称以"_s1.jpg"结尾的普通文件，如果其修改日期在一天内，则将其打包到/tmp/back.tar.gz文件中

```
find /tmp -type f -name ".*_sj.jpg" -mtime 1|xarges tar zxf
/tmp/back.tar.gz
```

72、配置mysql服务器的时候，配置了auto_increment_increment=3，请问这里的3意味着什么?

```
auto_increment是用于主键自动增长的，从3开始增长，3表示自增的起始值
```

73、详细说明keepalived的故障切换工作原理

这种故障切换是通过VRRP协议来实现的，主节点会按一定的时间间隔发送心跳信息的广播包，告诉备节点自己的存活状态信息，当主节点发生故障时，备节点在一段时间内就收到广播包，从而判断主节点出现故障，因此会调用自身的接管程序来接管主节点的IP资源及服务，当主节点恢复时，备节点会主动释放资源，恢复到接管前的状态，从而来实现主备故障切换

74、什么是系统调用?

根据进程访问资源的特点，可以把进程在系统上的运行分为两个级别：

用户态(user mode) : 用户态运行的进程或可以直接读取用户程序的数据。

系统态(kernel mode):可以简单的理解系统态运行的进程或程序几乎可以访问计算机的任何资源，不受限制。

说了用户态和系统态之后，那么什么是系统调用呢?
运行的应用程序基本都是运行在用户态，如果调用操作系统提供的系统态级别的子功能咋办呢？那就需要系统调用了!

也就是说在运行的用户程序中，凡是与系统态级别的资源有关的操作（如文件管理、进程控制、内存管理等)，都必须通过系统调用方式向操作系统提出服务请求，并由操作系统代为完成。

这些系统调用按功能大致可分为如下几类：

- 设备管理。完成设备的请求或释放，以及设备启动等功能。
- 文件管理。完成文件的读、写、创建及删除等功能。
- 进程控制。完成进程的创建、撤销、阻塞及唤醒等功能。
- 进程通信。完成进程之间的消息传递或信号传递等功能。
- 内存管理。完成内存的分配、回收以及获取作业占用内存区大小及地址等功能。

75、进程和线程的区别?

线程是进程划分成的更小的运行单位,一个进程在其执行的过程中可以产生多个线程。线程和进程最大的不同在于基本上各进程是独立的，而各线程则不一定，因为同一进程中的线程极有可能会相互影响。线程执行开销小，但不利于资源的管理和保护；而进程正相反。

76、进程有哪几种状态?

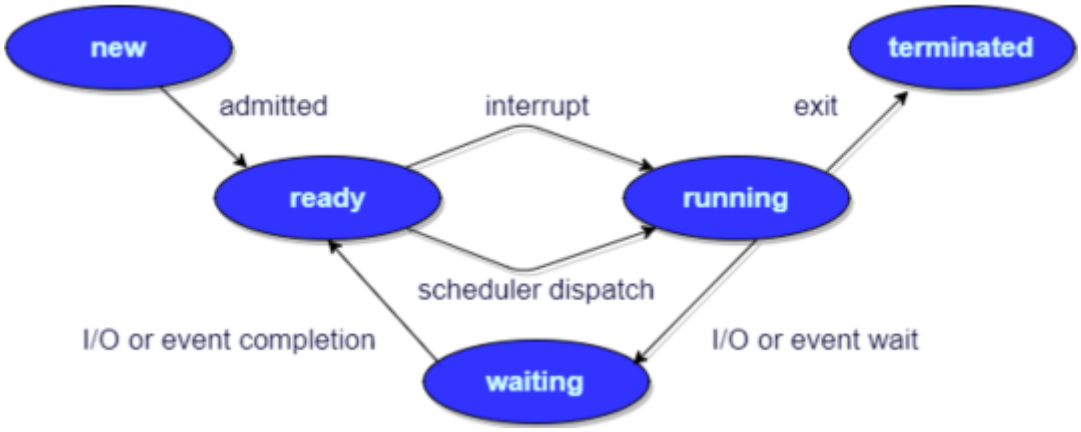
创建状态(new)： 进程正在被创建，尚未到就绪状态。

就绪状态(ready)： 进程已处于准备运行状态，即进程获得了除了处理器之外的一切所需资源，一旦得到处理器资源(处理器分配的时间片)即可运行。

运行状态(running)： 进程正在处理器上上运行(单核 CPU 下任意时刻只有一个进程处于运行状态)。

阻塞状态(waiting)： 又称为等待状态，进程正在等待某一事件而暂停运行如等待某资源为可用或等待 IO 操作完成。即使处理器空闲，该进程也不能运行。

结束状态(terminated)： 进程正在从系统中消失。可能是进程正常结束或其他原因中断退出运行。



77、进程间的通信方式

管道/匿名管道(Pipes)： 用于具有亲缘关系的父子进程间或者兄弟进程之间的通信。

有名管道(Names Pipes) : 匿名管道由于没有名字，只能用于亲缘关系的进程间通信。为了克服这个缺点，提出了有名管道。有名管道严格遵循先进先出(first in first out)。有名管道以磁盘文件的方式存在，可以实现本机任意两个进程通信。

信号(Signal)： 信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生;

消息队列(Message Queuing)： 消息队列是消息的链表,具有特定的格式,存放在内存中

并由消息队列标识符标识。管道和消息队列的通信数据都是先进先出的原则。与管道（无名管道：只存在于内存中的文件；命名管道：存在于实际的磁盘介质或者文件系统）不同的是消息队列存放在内核中，只有在内核重启(即，操作系统重启)或者显示地删除一个消息队列时，该消息队列才会被真正的删除。消息队列可以实现消息的随机查询,消息不一定要以先进先出的次序读取,也可以按消息的类型读取.比 FIFO 更有优势。消息队列克服了信号承载信息量少，管道只能承载无格式字 节流以及缓冲区大小受限等缺。

信号量(Semaphores)： 信号量是一个计数器，用于多进程对共享数据的访问，信号量的意图在于进程间同步。这种通信方式主要用于解决与同步相关的问题并避免竞争条件。

共享内存(Shared memory)： 使得多个进程可以访问同一块内存空间，不同进程可以及时看到对方进程中对共享内存中数据的更新。这种方式需要依靠某种同步操作，如互斥锁和信号量等。可以说这是最有用的进程间通信方式。

套接字(Sockets) : 此方法主要用于在客户端和服务器之间通过网络进行通信。套接字是支持 TCP/IP 的网络通信的基本操作单元，可以看做是不同主机之间的进程进行双向通信的端点，简单的说就是通信的两方的一种约定，用套接字中的相关函数来完成通信过程。

78、线程间的同步的方式

线程同步是两个或多个共享关键资源的线程的并发执行。应该同步线程以避免关键的资源使用冲突。操作系统一般有下面三种线程同步的方式：

互斥量(Mutex)：采用互斥对象机制，只有拥有互斥对象的线程才有访问公共资源的权限。因为互斥对象只有一个，所以可以保证公共资源不会被多个线程同时访问。比如 Java 中的 synchronized 关键词和各种 Lock 都是这种机制。

信号量(Semphares)： 它允许同一时刻多个线程访问同一资源，但是需要控制同一时刻访问此资源的最大线程数量

事件(Event) :Wait/Notify：通过通知操作的方式来保持多线程同步，还可以方便的实现多线程优先级的比较操作

79、进程的调度算法

先到先服务(FCFS)调度算法 : 从就绪队列中选择一个最先进入该队列的进程为之分配资源，使它立即执行并一直执行到完成或发生某事件而被阻塞放弃占用 CPU 时再重新调度。

短作业优先(SJF)的调度算法 : 从就绪队列中选出一个估计运行时间最短的进程为之分配资源，使它立即执行并一直执行到完成或发生某事件而被阻塞放弃占用 CPU 时再重新调度。

时间片轮转调度算法 : 时间片轮转调度是一种最古老，最简单，最公平且使用最广的算法，又称 RR(Round robin)调度。每个进程被分配一个时间段，称作它的时间片，即该进程允许运行的时间。

多级反馈队列调度算法： 前面介绍的几种进程调度的算法都有一定的局限性。如短进程优先的调度算法，仅照顾了短进程而忽略了长进程 。多级反馈队列调度算法既能使高优先级的作业得到响应又能使短作业（进程）迅速完成。，因而它是目前被公认的一种较好的进程调度算法，UNIX 操作系统采取的便是这种调度算法。

优先级调度： 为每个流程分配优先级，首先执行具有最高优先级的进程，依此类推。具有相同优先级的进程以 FCFS 方式执行。可以根据内存要求，时间要求或任何其他资源要求来确定优先级。

80、操作系统的内存管理主要是做什么?

操作系统的内存管理主要负责内存的分配与回收（malloc 函数：申请内存，free 函数：释放内存），另外地址转换也就是将逻辑地址转换成相应的物理地址等功能也是操作系统内存管理做的事情。

81、常见的几种内存管理机制

简单分为连续分配管理方式和非连续分配管理方式这两种。连续分配管理方式是指为一个用户程序分配一个连续的内存空间，常见的如 块式管理 。同样地，非连续分配管理方式允许一个程序使用的内存分布在离散或者说不相邻的内存中，常见的如页式管理 和 段式管理。

块式管理： 远古时代的计算机操系统的内存管理方式。将内存分为几个固定大小的块，每个块中只包含一个进程。如果程序运行需要内存的话，操作系统就分配给它一块，如果程序运行只需要很小的空间的话，分配的这块内存很大一部分几乎被浪费了。这些在每个块中未被利用的空间，称之为碎片。

页式管理： 把主存分为大小相等且固定的一页一页的形式，页较小，相对相比于块式管理的划分力度更大，提高了内存利用率，减少了碎片。页式管理通过页表对应逻辑地址和物理地址。

段式管理： 页式管理虽然提高了内存利用率，但是页式管理其中的页实际并无任何实际意义。 段式管理把主存分为一段段的，每一段的空间又要比一页的空间小很多 。但是，

最重要的是段是有实际意义的，每个段定义了一组逻辑信息，例如,有主程序段 MAIN、子程序段 X、数据段 D 及栈段 S 等。 段式管理通过段表对应逻辑地址和物理地址。

82、快表和多级页表

在分页内存管理中，很重要的两点是：

虚拟地址到物理地址的转换要快。

解决虚拟地址空间大，页表也会很大的问题。

快表

为了解决虚拟地址到物理地址的转换速度，操作系统在 页表方案 基础之上引入了 快表 来加速虚拟地址到物理地址的转换。可以把块表理解为一种特殊的高速缓冲存储器（Cache），其中的内容是页表的一部分或者全部内容。作为页表的 Cache，它的作用与页表相似，但是提高了访问速率。由于采用页表做地址转换，读写内存数据时 CPU 要访问两次主存。有了快表，有时只要访问一次高速缓冲存储器，一次主存，这样可加速查找并提高指令执行速度。

使用快表之后的地址转换流程是这样的：

根据虚拟地址中的页号查快表；

如果该页在快表中，直接从快表中读取相应的物理地址；

如果该页不在快表中，就访问内存中的页表，再从页表中得到物理地址，同时将页表中的该映射表项添加到快表中；

当快表填满后，又要登记新页时，就按照一定的淘汰策略淘汰掉快表中的一个页。

看完了之后会发现快表和平时经常在开发的系统使用的缓存（比如 Redis）很像，的确是这样的，操作系统中的很多思想、很多经典的算法，都可以在日常开发使用的各种工具或者框架中找到它们的影子。

多级页表

引入多级页表的主要目的是为了避免把全部页表一直放在内存中占用过多空间，特别是那些根本就不需要的页表就不需要保留在内存中。多级页表属于时间换空间的典型场景。

83、分页机制和分段机制的共同点和区别

共同点：

分页机制和分段机制都是为了提高内存利用率，较少内存碎片。

页和段都是离散存储的，所以两者都是离散分配内存的方式。但是，每个页和段中的内存是连续的。

区别：

页的大小是固定的，由操作系统决定；而段的大小不固定，取决于当前运行的程序。

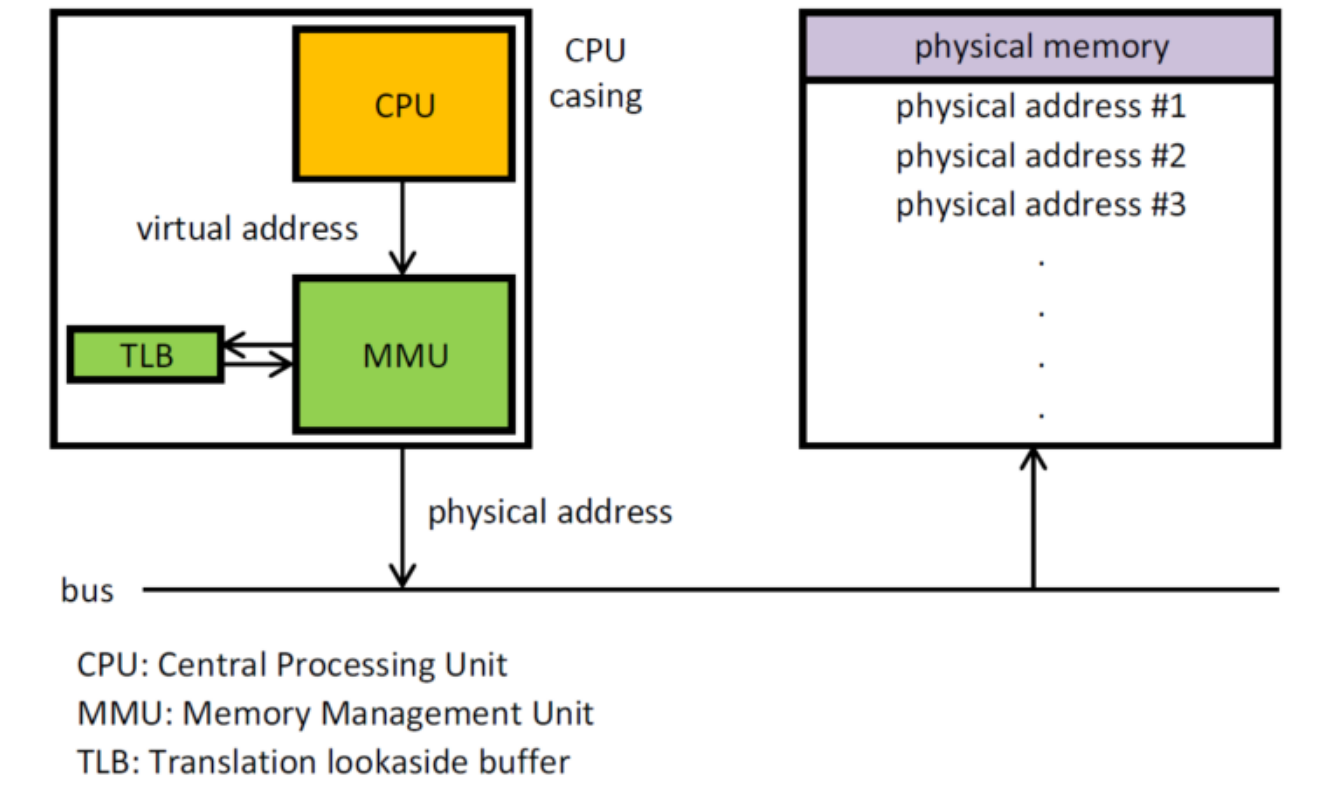
分页仅仅是为了满足操作系统内存管理的需求，而段是逻辑信息的单位，在程序中可以体现为代码段，数据段，能够更好满足用户的需要。

84、逻辑(虚拟)地址和物理地址

逻辑地址由操作系统决定。物理地址指的是真实物理内存中地址，更具体一点来说就是内存地址寄存器中的地址。物理地址是内存单元真正的地址。

85、CPU 寻址了解吗?为什么需要虚拟地址空间?

现代处理器使用的是一种称为 虚拟寻址(Virtual Addressing) 的寻址方式。使用虚拟寻址，CPU 需要将虚拟地址翻译成物理地址，这样才能访问到真实的物理内存。 实际上完成虚拟地址转换为物理地址转换的硬件是 CPU 中含有一个被称为 内存管理单元（Memory Management Unit, MMU） 的硬件。



为什么要有虚拟地址空间呢?

没有虚拟地址空间的时候，程序都是直接访问和操作的都是物理内存 。但是这样有什么问题呢?

用户程序可以访问任意内存，寻址内存的每个字节，这样就很容易（有意或者无意）破坏操作系统，造成操作系统崩溃。

想要同时运行多个程序特别困难，比如想同时运行一个微信和一个 QQ 音乐都不行。为什么呢？举个简单的例子：微信在运行的时候给内存地址 1xxx 赋值后，QQ 音乐也同样给内存地址 1xxx 赋值，那么 QQ 音乐对内存的赋值就会覆盖微信之前所赋的值，这就造成了微信这个程序就会崩溃。

总结来说：如果直接把物理地址暴露出来的话会带来严重问题，比如可能对操作系统造成伤害以及给同时运行多个程序造成困难。

通过虚拟地址访问内存有以下优势：

程序可以使用一系列相邻的虚拟地址来访问物理内存中不相邻的大内存缓冲区。

程序可以使用一系列虚拟地址来访问大于可用物理内存的内存缓冲区。当物理内存的供应量变小时，内存管理器会将物理内存页（通常大小为 4 KB）保存到磁盘文件。数据或代码页会根据需要在物理内存与磁盘之间移动。
不同进程使用的虚拟地址彼此隔离。一个进程中的代码无法更改正在由另一进程或操作系统使用的物理内存。

86、什么是虚拟内存(Virtual Memory)?

虚拟内存是计算机系统内存管理的一种技术，可以手动设置自己电脑的虚拟内存。不要单纯认为虚拟内存只是“使用硬盘空间来扩展内存“的技术。虚拟内存的重要意义是它定义了一个连续的虚拟地址空间，并且 把内存扩展到硬盘空间。

87、局部性原理

局部性原理表现在以下两个方面：

时间局部性 ： 如果程序中的某条指令一旦执行，不久以后该指令可能再次执行；如果某数据被访问过，不久以后该数据可能再次被访问。产生时间局部性的典型原因，是由于在程序中存在着大量的循环操作。

空间局部性 ： 一旦程序访问了某个存储单元，在不久之后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址，可能集中在一定的范围之内，这是因为指令通

常是顺序存放、顺序执行的，数据也一般是以向量、数组、表等形式簇聚存储的。

时间局部性是通过将近来使用的指令和数据保存到高速缓存存储器中，并使用高速缓存的层次结构实现。空间局部性通常是使用较大的高速缓存，并将预取机制集成到高速缓存控制逻辑中实现。虚拟内存技术实际上就是建立了 “内存—外存”的两级存储器的结构，利用局部性原理实现高速缓存。

88、虚拟存储器

基于局部性原理，在程序装入时，可以将程序的一部分装入内存，而将其他部分留在外存，就可以启动程序执行。由于外存往往比内存大很多，所以运行的软件的内存大小实际上是可以比计算机系统实际的内存大小大的。在程序执行过程中，当所访问的信息不在内存时，由操作系统将所需要的部分调入内存，然后继续执行程序。另一方面，操作系统将内存中暂时不使用的内容换到外存上，从而腾出空间存放将要调入内存的信息。这样，计算机好像为用户提供了一个比实际内存大的多的存储器——虚拟存储器。
实际上，虚拟内存同样是一种时间换空间的策略，用 CPU 的计算时间，页的调入调出花费的时间，换来了一个虚拟的更大的空间来支持程序的运行。程序世界几乎不是时间换空间就是空间换时间。

89、虚拟内存的技术实现

虚拟内存的实现需要建立在离散分配的内存管理方式的基础上。 虚拟内存的实现有以下三种方式：

请求分页存储管理 ： 建立在分页管理之上，为了支持虚拟存储器功能而增加了请求调页功能和页面置换功能。请求分页是目前最常用的一种实现虚拟存储器的方法。请求分页存储管理系统中，在作业开始运行之前，仅装入当前要执行的部分段即可运行。假如在作业运行的过程中发现要访问的页面不在内存，则由处理器通知操作系统按照对应的页面置换算法将相应的页面调入到主存，同时操作系统也可以将暂时不用的页面置换到外存中。

请求分段存储管理 ： 建立在分段存储管理之上，增加了请求调段功能、分段置换功能。请求分段储存管理方式就如同请求分页储存管理方式一样，在作业开始运行之前，仅装入当前要执行的部分段即可运行；在执行过程中，可使用请求调入中断动态装入要访问但又不在内存的程序段；当内存空间已满，而又需要装入新的段时，根据置换功能适当调出某个段，以便腾出空间而装入新的段。

请求段页式存储管理

不管是上面那种实现方式，一般都需要：

一定容量的内存和外存：在载入程序的时候，只需要将程序的一部分装入内存，而将其

他部分留在外存，然后程序就可以执行了；

缺页中断：如果需执行的指令或访问的数据尚未在内存（称为缺页或缺段），则由处理器通知操作系统将相应的页面或段调入到内存，然后继续执行程序；

虚拟地址空间： 逻辑地址到物理地址的变换。

90、页面置换算法

地址映射过程中，若在页面中发现所要访问的页面不在内存中，则发生缺页中断 。

缺页中断 就是要访问的页不在主存，需要操作系统将其调入主存后再进行访问。 在这个时候，被内存映射的文件实际上成了一个分页交换文件。

当发生缺页中断时，如果当前内存中并没有空闲的页面，操作系统就必须在内存选择一个页面将其移出内存，以便为即将调入的页面让出空间。用来选择淘汰哪一页的规则叫做页面置换算法，可以把页面置换算法看成是淘汰页面的规则。
OPT 页面置换算法（最佳页面置换算法）： 最佳(Optimal, OPT)置换算法所选择的被淘汰页面将是以后永不使用的，或者是在最长时间内不再被访问的页面,这样可以保证获得最低的缺页率。但由于人们目前无法预知进程在内存下的若干页面中哪个是未来最长时间内不再被访问的，因而该算法无法实现。一般作为衡量其他置换算法的方法。

FIFO（First In First Out） 页面置换算法（先进先出页面置换算法）：总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面进行淘汰。

LRU （Least Currently Used）页面置换算法（最近最久未使用页面置换算法）： LRU 算法赋予每个页面一个访问字段，用来记录一个页面自上次被访问以来所经历的时间 T，当须淘汰一个页面时，选择现有页面中其 T 值最大的，即最近最久未使用的页面予以淘汰。

LFU （Least Frequently Used）页面置换算法（最少使用页面置换算法）：该置换算法选择在之前时期使用最少的页面作为淘汰页。