[1、目的 2](#_Toc24830059)

[2、计算机操作系统的发展（包括常用的操作系统） 2](#_Toc24830060)

[2.1 手工操作(无操作系统) 2](#_Toc24830061)

[2.2 单道批处理系统 2](#_Toc24830062)

[2.3 多道批处理系统 2](#_Toc24830063)

[2.4 分时系统 2](#_Toc24830064)

[2.5 实时系统 3](#_Toc24830065)

[2.6 通用操作系统 3](#_Toc24830066)

[2.7 总结 3](#_Toc24830067)

[3、Linux操作系统 3](#_Toc24830068)

[3.1 Linux的应用领域 3](#_Toc24830069)

[3.2 Linux版本分支 4](#_Toc24830070)

[4、苹果操作系统(绑定硬件) 4](#_Toc24830071)

[5、Windows操作系统 4](#_Toc24830072)

[6、苹果与微软 4](#_Toc24830073)

[7、Linux系统的进程管理和进程通信 5](#_Toc24830074)

[7.1进程的引入 5](#_Toc24830075)

[7.2进程控制块PCB 5](#_Toc24830076)

[7.3进程属性 6](#_Toc24830077)

[7.4优先级问题 6](#_Toc24830078)

[7.5进程调度算法简介 6](#_Toc24830079)

[7.6 linux下进程间通信的几种主要手段简介： 7](#_Toc24830080)

[8、个人见解 8](#_Toc24830081)

[8.1概况 8](#_Toc24830082)

[8.2第一阶段 8](#_Toc24830083)

[8.3第二阶段 8](#_Toc24830084)

[8.4 对Linux进程管理和进程通信的理解 8](#_Toc24830085)

[8.5 Linux下进程通信的八种方法： 9](#_Toc24830086)

# 资料检索作业

## 1、目的

通过网络搜索，尽可能多地了解当前操作系统的发展和常用的操作系统，了解Linux系

统的进程管理和进程通信方法。

## 2、计算机操作系统的发展（包括常用的操作系统）

### 2.1 手工操作(无操作系统)

 特点：用户独占机器，CPU等待工作，CPU利用不充分

### 2.2 单道批处理系统

加载一个监督软件，自动成批的处理或执行一个或多个用户的作业

2.2.1 联机批处理系统

作业的输入输出由CPU决定，在主机和输入机之间增加一个磁带机，在监督程序的控制下，计算机自动完成任务

缺点：在输入作业和输出结果时,CPU还会处于空闲状态，等待输入/输出设备完成工作--主机处于忙等状态

2.2.2 脱机批处理系统

为了克服CPU与输入输出设备速度不匹配的问题，提高CPU利用率，输入输出脱离主机控制

特点：增加一台不和主机相连的卫星机

### 2.3 多道批处理系统

在批处理系统后，引入多道程序设计技术后形成多道批处理系统，允许多个程序同时进入内存并运行，允许他们在CPU

中交替执行，共享系统中的所有资源

缺点：不提供人机交互能力，给用户使用计算机带来不便

### 2.4 分时系统

把处理机的时间分成很短的时间片，把时间片轮流分配给各个作业使用，由于运算速度快，给每个

用户的感觉就是独占一台计算机，允许每个用户在充分的人机交互的情况下，完成作业的运行。

缺点：无法对特殊任务做出及时响应

### 2.5 实时系统

系统能够及时相应外部发生的事件，并在严格时间完成对该事件的处理

2.5.1 实时控制系统

飞机飞行，导弹发射等自动控制系统；轧钢，石化等工业生产过程总的自动控制系统

2.5.2 实时信息处理系统

预定飞机票，查询航班；银行系统，情报检索系统。

特点：及时响应，高可靠性

### 2.6 通用操作系统

操作系统的三种基本类型：多道批处理系统，分时系统，实时系统

定义：具有多种类型操作特征的操作系统，同时具有多道批处理系统，分时，实时的功能或者两种以上的功能

### 2.7 总结

以上的这些操作系统都是在20世纪50年代到60年代间发展的，现在我们都看不到了，只能去想象了。

之后能留给我们最老的操作系统就是——UNIX操作系统(1973年发布)

UNIX被称为计算机/互联网行业的基石，后来的很多操作系统都是在他的基础上发展而来的(linux macos)

由于其优秀的稳定性，现在在大的金融公司还依旧使用该系统，重要的一点UNIX和硬件绑定

## 3、Linux操作系统

1991年 Linus在研究Minix的基础上，在论坛上宣布Linux内核系统的诞生，称作为freex,后来改名为Linux

### 3.1 Linux的应用领域

企业服务器-企业

嵌入式-手机，航空航天等领域

桌面-个人电脑

其他

### 3.2 Linux版本分支

3.2.1 debian分支(deb包管理)

Ubuntu

kali

麒麟

3.2.2 redhat分支(rpm包管理)

Red Hat Enterprise Linux (收费版本，提供技术支持，提供官方的下载更新服务器)

centos(rhel的社区克隆版本，免费，由第三方的企业提供下载更新) (之后被红帽公司收购的，可控之内发展)

Fedora Core(由Red Hat桌面版发展而来 ，免费)

3.2.3 slackware分支

suse

opensuse(免费)

3.2.4 Android

2003年 ，安迪 鲁宾创办了Android公司，基于Linux的开放移动操作系统

2005年，Android公司被Goole手稿

2007年，谷歌公司发布Android操作系统

## 4、苹果操作系统(绑定硬件)

1984年，苹果发布了System1 黑白界面的操作系统(基于Unix)

2000年，苹果发不了 Mac Os x的公开操作系统测试版

2003年 Mac Os x 10.3正式上市

## 5、Windows操作系统

1985年，Microsoft-Dos模拟环境

1995年，Microsoft公司推出Windows-95

1998年，Microsoft公司推出Windows-98

## 6、苹果与微软

1973年施乐公司开发除了Alto

1979年乔布斯模仿，后来开发Macintosh

1980年微软和IBM合作，预装DOS

1981年 乔布斯邀请盖茨看Macintosh样机，盖茨震惊，从施乐挖人

1984年Macintosh发布，风靡世界

1985年windows1.0发布，乔布斯告盖茨抄袭苹果

## 7、Linux系统的进程管理和进程通信

### 7.1进程的引入

当计算机在引入多道程序时，出现了临界资源竞争的情况，为了刻画和解决程序间的这种制约关系，提出了进程的概念，用以改善资源的利用率，提高程序的吞吐量。

### 7.2进程控制块PCB

linux系统的所有进程控制块都是通过结构体指针数组形式的数据结构来表示的(每个PCB块大约有1kb)：

/\*全局变量nr\_task动态记录进程数\*/

struct task\_sturct \*task[NR\_TASK]={&init\_task};

运行状态进程的PCB同样用结构体指针数组表示:

current\_set[];

task\_struct结构体定义：

struct task\_struct{

...

unsigned short uid;

int pid;

...

volatile long state;

long prority;

unsigned long rt\_protity;

long counter;

unsigned long policy;

...

struct task\_struct \*next\_task, \*prev\_task;

struct task\_struct \*next\_run, \*prev\_run;

struct task+struct \*p\_opptr, \*pptr, \*p\_cptr, \*pysptr, \*p\_ptr;

...

};

***数据成员说明：***  
**uid：**用户标识；  
**pid：**进程标识；  
**state：**进程状态标识；  
1）可运行状态  
2）可中断阻塞状态  
3）不可中断阻塞状态  
4）僵死状态  
5）暂停状态  
6）交换状态  
**protity：**进程优先级；  
**counter：**优先级计数器，用于进程轮转调度算法；  
**policy：**进程调度策略；  
**next\_task,prev\_task：**PCB双向链表指针；  
**p\_opptr ...：**指明进程家族间的关系，分别为祖父进程，父进程，子进程，以及新老进程指针；

### 7.3进程属性

进程ID：PID，进程唯一标志符；

父进程ID：PPID；

启动进程的用户ID：UID；

所归属的组ID：GID；

进程执行的优先级；

进程连接终端

### 7.4优先级问题

**谦让度：**标识进程之间资源竞争能力，高谦让度的进程，资源竞争能力弱。负值或0表示最高优先级，对其他进程不谦让。谦让度的值一般为：-20～19；当前硬件发展速度非常快，一般我们是不设置进程优先级的，除非有进程失控而疯狂的抢占资源；

在创建进程时，nice可以为进程设置谦让值，但进程优先级的值为父进程shell优先级的值与nice所指定的谦让度相加的结果。即，利用nice指定的值是一个增量，而不是优先级的绝对值。

运行 a.out 程序，并为它指定谦让度增量为5：

nice -n 5 a.out &

利用renice来更改谦让度：

renice 谦让度 PID

### 7.5进程调度算法简介

程序使用cpu的三种模式：I／O密集型、计算密集型和平衡型。对于I／O密集型程序，响应时间非常重要，响应时间不及时，和可能丢失数据；对于计算密集型程序，cpu的周转时间非常重要；对于平衡型程序，协调响应和cpu周转的工作最重要。

**进程周转时间：**等待进入内存的时间,在就绪队列中等待的时间,在 CPU中执行的时间和I/O操作的时间的总和。

**FCFS算法**

FCFS（frist come first serve），也叫FIFO算法，先来先执行。短任务执行可能会非常慢，因为前面的进程可能占用很长时间，造成平均响应时间过长。

**时间片轮转算法**

目的是改善短程序响应时间长的问题。方法是周期性的对进程进行切换。该算法最关键的时间片的选择，时间片过长和FIFO算法区别不大，时间片过短，进程频繁切换的开销大于执行程序的开销，系统效率降低。

**STCF算法**

STCF算法全称：short time to complete frist。核心是每个进程都有优先级，短进程的优先级要高于长进程的优先级。STCF算法分为：非抢占式和抢占式两类。非抢占式就是让已经在cpu上运行的程序执行到结束或者阻塞，然后在就绪队列中选择执行时间最短的程序来执行。而抢占式是，每当有新进程进入就绪队列时，都会去判断当前运行进程和新进程的执行时间长短，选择执行时间短的进程运行，及永远保证cpu运行的是所有进程中执行时间最短的进程。该算法有个非常明显的缺点：长任务进程无法执行。还有该算法的关键点是如何预测进程所需的执行时间。

**优先级调度算法**

解决了STCF算法中长任务进程饥饿的问题，而暴露出的短任务会饥饿的问题，可以通过动态调整优先级来解决。实际上动态调整优先级（称为权值）+ 时间片轮转的策略正是Linux系统进程调度的策略之一的分时调度策略，调度过程如下：

(1) 创建任务指定采用分时调度策略，指定优先级nice的值（-20～19）

(2) 依据nice的值确定在cpu上的执行时间

(3) 如果没有等待资源，则将该任务加入就绪队列中

(4) 调度程序遍历就绪队列，通过对动态优先级的计算，选择计算结果最

大的去执行。当时间片用完时或进程主动放弃cpu时，该任务就重新放回就

绪队列队尾或等待队列中。

(5) 调度程序重新按照（4）的方式调度进程。

(6) 当调度程序发现所有就绪队列中的进程的权值都不大于0时，重复(2)

的过程

现代OS通常都会采用**混合调度算法**。如：将不同的进程分为几个大类，每个大类有不同的优先级，不同大类的进程调度取决于大类的优先级，同一个大类的进程调度采用时间片轮转算法来实现。

### 7.6 linux下进程间通信的几种主要手段简介：

管道（Pipe）及有名管道（named pipe）：管道可用于具有亲缘关系进程间的通信，有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；

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）；

报文（Message）队列（消息队列）：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点。

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。

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。

套接口（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。

## 8、个人见解

### 8.1概况

俗话说读史使人明智，纵览操作系统的发展历史（总的来说，分为两个阶段），我也是对计算机有了更深的认识：

### 8.2第一阶段

计算机最开始的使用原则只是专注于计算，所以刚出来的批处理操作系统便是如此；随后需要计算机干多件事，即多道程序，作业之间的切换和调度成为核心，因此，多进程结构和进程管理概念开始萌芽；到后来，计算机的使用人数逐渐增多，如果每一个人启动一个作业，作业之间的人数增加，所以就出现了分时系统；不久后，又出现了小型计算机（UNIX），越来越多的人可以使用计算机；终于，个人计算机开始普及（Linux），很多人可以使用计算机并接触（UNIX）。

上述过程（用当时的典型代表表示）：IBSYS->OS/360->MULTICS->Unix->Linux

### 8.3第二阶段

自从IBM推出PC机后，自然也配套的装上了一个操作系统（CP/M），随后在此基础上又开发了QDOS；后来，比尔盖茨开创了微软，并买下来QDOS，并改名为MS-DOS，MS-DOS的磁盘、文件、命令使用方便，但似乎可以更方便；后来，MS-DOS支持了鼠标和键盘，微软也将放弃MS-DOS，不久后Windows 3.0大获成功，后来就是一发不可收拾了，95，XP，Vista，Win 7，8，9，到现在我们大部分人用的Windows10

另外，苹果推出来PC机，简称Mac机，其操作系统为System 7（后改名为Mac OS 8）

Mac OS核心是Unix，专注于界面、文件、媒体等和用户有关的内容。

上述过程（用当时的典型代表表示）：CP/M ->QDOS->MS-DOS-> Windows

### 8.4 对Linux进程管理和进程通信的理解

在Linux系统中，把每个正在执行的程序成为一个进程。每个进程都分配一个ID号；每个进程都会对应一个父进程，而这个父进程可以赋值多个字进程；每个进程都以两种方式存在，前台与后台；一般系统的服务都是以后台进程的方式存在，而且都会常驻在系统中，知道关机才结束。常用的进程管理了命令为ps，top，pstree，kill等：

执行格式： # ps [-auf]

-a 列出当前终端上启动的所有进程；

-u 按用户名和启动时间的顺序来显示进程；

-e列出系统中所有进程；

-f 以全格式列出；

-l以long格式列出；

执行格式： # top 每隔5s统计显示一次系统中的进程状态

-q 不断地更新、显示系统中的进程状态；

-d 指定每两次屏幕信息刷新之间的时间间隔。

执行格式: #pstree -h 列出进程树并高亮标出当前进程

此命令可展示出系统中的进程树结构。

其中，init是1号进程，系统所有进程都是它派生的。

执行格式：# kill [-9] PID （PID为利用ps命令所查出的process ID）

例： kill -9 456 终止process ID 为456的process

### 8.5 Linux下进程通信的八种方法：

管道(pipe)，

命名管道(FIFO)，

内存映射(mapped memeory)，

消息队列(message queue)，

共享内存(shared memory)，

信号量(semaphore)，

信号(signal)

套接字(Socket)