完整系统组成部分

用户应用层

操作系统服务层

操作系统内核层 与计算机硬件进行交互

硬件系统

单内核

调用服务的主程序层

执行系统调用的服务层

支持系统调用的底层函数

## Linux内核体系结构：

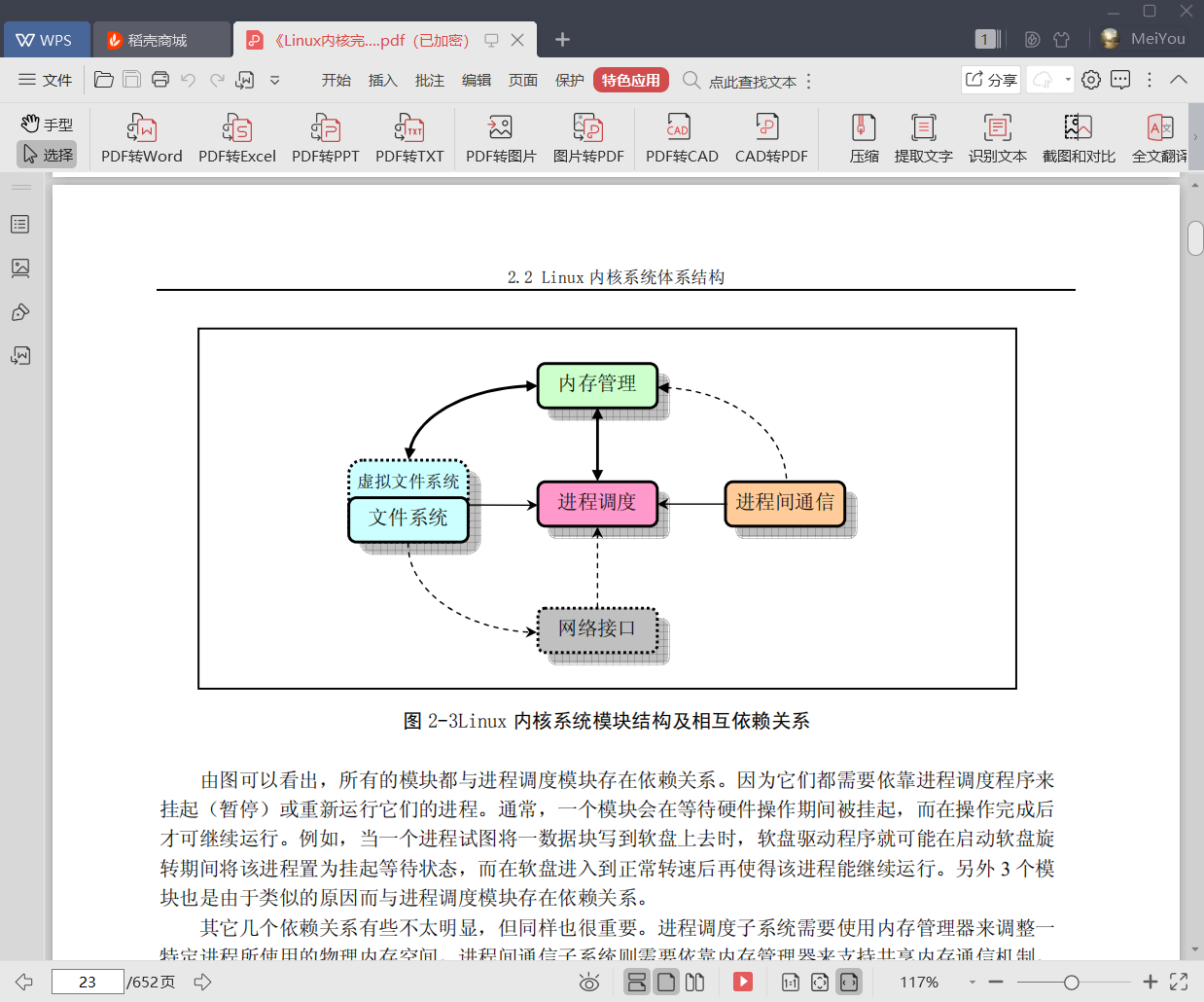
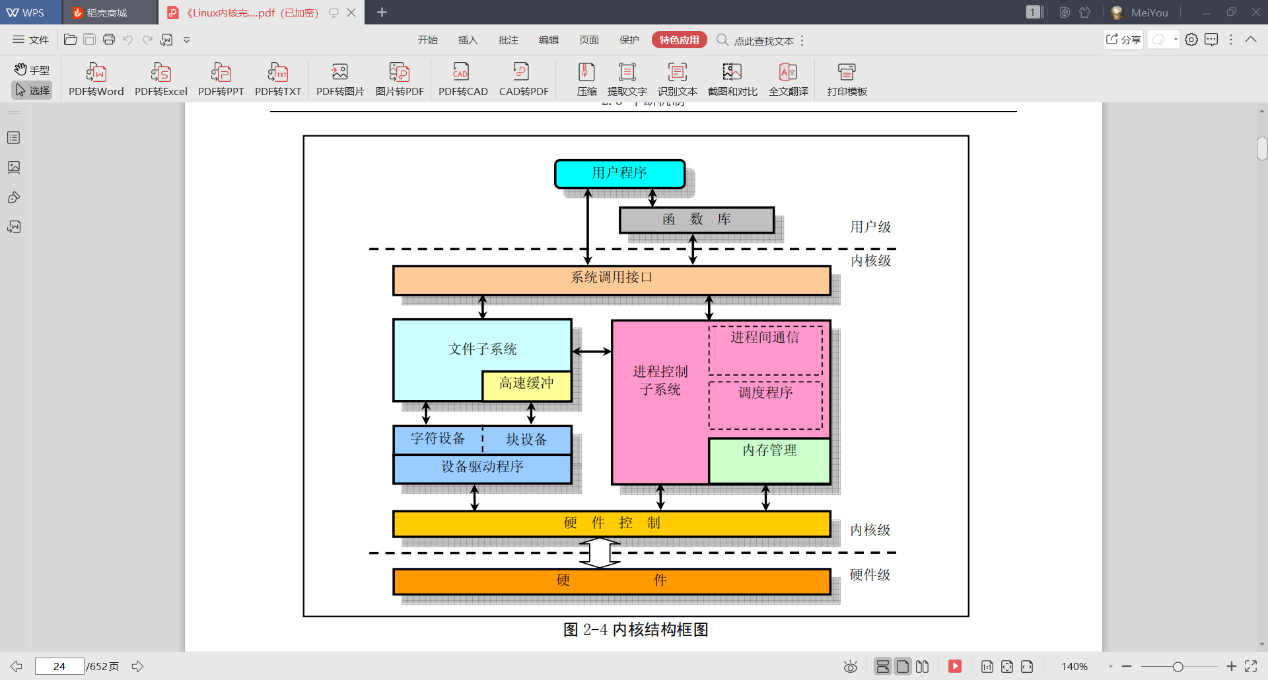
进程调度模块 控制进程对CPU的使用

内存管理模块 安全共享机器内存区，支持虚拟内存管理

文件系统模块 支持外部设备的驱动和存储，虚拟内存

进程通信模块 进程间通信模块子系统支持多种进程间的信息交换方式

网络接口模块 网络接口模块提供多种网络通信标准的访问和支持多种硬件



进程通信 进程调度 内存管理 进程控制子系统中

文件系统模块 文件子系统

## 中断机制

***内核和硬件的通信方式：***

***中断***

中断是指在CPU正常运行期间，由于内外部事件或由程序预先安排的事件引起的CPU暂时停止正在运行的程序，转而处理外部事件或者程序事先安排的事件中去，服务完毕后再返回去继续运行被暂时中断的程序。

***轮询***

早期的CPU处理外设的事件(比如接收键盘输入)，往往采用“轮询”的方式。即CPU像个查岗的一样轮番对外设顺序访问，比如它先看看键盘有没被按下，有的话就处理，没的话继续往下看鼠标有没有移动，再看看打印机……这种方式使CPU的执行效率很低，CPU需要暂停，去巡查外设是否有输入，(此时CPU处于空闲状态)。

**中断模式的作用和优点在于：**

1. 可以使CPU和外设同时工作(外设等待CPU响应，而不是CPU等待外设，CPU速度远远快于外设)，使系统可以及时地响应外部事件。

2. 可允许多个外设同时工作，大大提高了CPU的利用率，也提高了数据输入、输出的速度，（允许多个外设等待CPU响应）。

3. 可以使CPU及时处理各种软硬件故障（比如计算机在运行过程中，出现了难以预料的情况或一些故障，如电源掉电、存储出错、运算溢出等等。计算机可以利用中断系统自行处理，而不必停机或报告工作人员。）

**中断类型**

**同步中断和异步中断**

1.同步中断是在指令执行时由CPU主动产生的，又称为内部中断，受到CPU控制，其执行点是可控的，这里同步是指中断请求信号与代码指令之间的同步执行，在一条指令执行完毕后，CPU才能进行中断，不能在执行期间。

——异常（exception）。

故障（fault）、陷阱（trap）、终止（abort）三类。

2.异步中断是CPU被动接收到的，由外部硬件设备产生，又称为外部中断，其发生时间不可预测, 包括指令执行期间

——中断（interrupt）。

中断可分为可屏蔽中断（Maskable Interrupt）和

非可屏蔽中断（Nomaskable Interrupt）

典型例子：

1. 典型的可屏蔽中断的例子是打印机中断，CPU对打印机中断请求的响应可以快一些，也可以慢一些，因为让打印机稍等待一会也是完全合理的。

　　2. 典型的非屏蔽中断的例子是电源断电，一旦出现此中断请求，必须立即无条件地响应，否则进行其他任何工作都是没有意义的。

从广义上讲，中断又可分为四类：中断、 故障、陷阱、终止。

类别 原因 异步/同步 返回行为

中断 来自I/O设备的信号 异步 总是返回到下一条指令

陷阱 有意的异常 同步 总是返回到下一条指令

故障 潜在可恢复的错误 同步 返回到当前指令

终止 不可恢复的错误 同步 不会返回

## 系统定时

PC机上的可编程定时芯片，每隔十秒发送始终中断，CPU时间片管理

## 进程控制

程序是一个可执行文件，进程process是可执行文件的一个实例。利用分时技术linux可以同时运行多个进程（分时技术，将CPU划分为多个时间片，分布给各个进程，单个时间里只有一个进程，但相当于多个进程同时运行。

Linux0.11内核来说，可以同时具有64个进程，除了第一个是手工建立，其余都是进程利用fork创建的新进程，被创建的进程称为子进程（child process）创建者称为父进程（parent process）内核程序使用进程标识号来标识每个进程（process ID，pid），进程由指令代码、数据和堆栈区组成。进程中的代码和数据部分分别对应着一个执行文件的代码段和数据段。

每个进程只能执行自己代码和访问自己的数据段和堆栈区。进程之间的互相通信需要通过系统调用来进行。对于只有一个CPU的系统，在某一时刻只有一个程序在运行。内核通过进程调度程序控制各个进程运行

Linux中一个进程可以是（kernel mode）内核态或者（user mode）用户态下执行，内核堆栈和用户堆栈是分开的