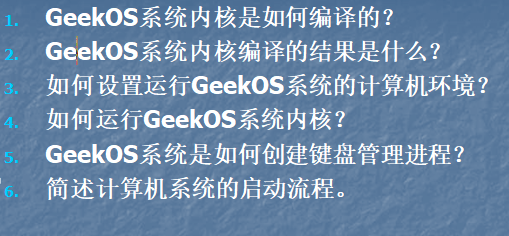
**Project0：**



1.GeekOS系统内核是如何编译的？

首先进入相应geekos版本的src的对应的project文件夹下的build目录下（geekos-0.3.0/src/project0/build目录），然后执行 make depend命令，通过扫描该目录下的所有C\C++ 代码，从而判断出文件之间的依赖关系，生成相应的makefile的规则，最后执行 make，根据makefile制定的规则，将c\c++文件编译成\*.o文件，然后进一步生成可执行文件。

2.GeekOS系统内核编译的结果是什么？

在build 目录下生成一个软盘镜像文件fd.img文件。

3.如何设置运行GeekOS系统的计算机环境？

首先安装ubuntu9，接着下载GeekOS，然后安装c/c++编译环境，执行命令sudo apt-get install gcc-3.4来安装gcc3.3，然后执行命令sudo rm gcc删除系统中原有的gcc连接，然后执行命令sudo ln –s gcc-3.4 gcc来修改gcc连接。接着需要执行执行命令sudo apt-get install bochs和sudo apt-get install bochs-x安装Bochs。

4.如何运行GeekOS系统内核？

调用Setup\_Kernel\_Thread函数建立一个待运行的内核线程。

5.GeekOS系统是如何创建键盘管理进程？

当用户进程需要从键盘输入信息，可调用Wait\_For\_Key()函数，该函数中首先通过信号量gotKey检查键盘缓冲区是否有按键，若gotKey为1，则说明此时键盘缓冲区有按键不为空，并从缓冲区取出一个键盘码，若gotKey为0，则说明此时键盘缓冲区中没有按键，则唤醒等待队列s\_waitQueue，并把进程放入该等待队列中。由于按键触发了键盘中断，键盘中断处理函数Keyboard\_Interrupt\_Handler中就会读取用户按键，将低级键扫描码转为含ASCII字符的高级代码，并刷新键盘缓冲区，最后唤醒等待队列s\_waitQueue中的等待按键进程继续执行。

6.简述计算机系统的启动流程。

计算机启动流程分为四个阶段：

第一阶段：BIOS阶段；即开机硬件自检、选择外部启动设备；

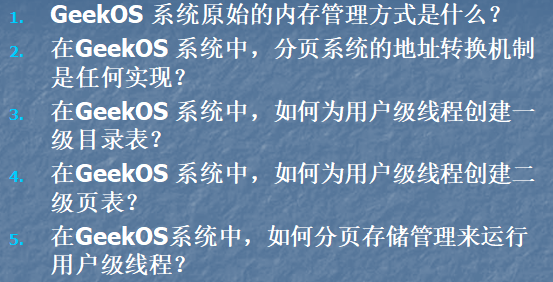
第二阶段：确定激活分区；即选择好外部设备后，BIOS通过依次遍历四个主分区找到激活分区；

第三阶段：确定操作系统的位置；即确定激活分区后，激活分区的第一个扇区为卷引导记录；

第四阶段：加载操作系统内核到内存中；通过卷引导记录找到操作系统内核的位置后，就可以加载内核进入内存。

（设备的前512KB存储的是主引导记录（MBR Master Boot Record），MBR里面存储了该设备分区表，最多有4个主分区，但是仅仅有一个主分区为**激活分区**，可以有一个**主分区**为**扩展分区**，**扩展分区**里面又可以划分无数个**逻辑分区**；）

**Project4**



1、GeekOS 系统原始的内存管理方式是什么？

答：基于段式的内存管理。

GeekOS的存储器管理：

①分页分配方式：系统中所有存储器都分成大小相等的块，称为页。在X86系统中，页的大小是4KB。

②堆分配方式：堆分配提供不同大小存储块的分配，使用函数Malloc（）和Free（）进行存储块的分配和回收。

2、在GeekOS 系统中，分页系统的地址转换机制是如何实现？

答：分页机制把线性地址空间和物理地址空间分别划分为大小相同的块。这样的块称之为页。通过在线性地址空间的页与物理地址空间的页之间建立的映射，分页机制实现线性地址到物理地址的转换。线性地址空间的页与物理地址空间的页之间的映射可根据需要而确定，可根据需要而改变。线性地址空间的任何一页，可以映射为物理地址空间中的任何一页。

  采用分页管理机制实现线性地址到物理地址转换映射的主要目的是便于实现虚拟存储器。不像段的大小可变，页的大小是相等并固定的。根据程序的逻辑划分段，而根据实现虚拟存储器的方便划分页。

3、在GeekOS 系统中，如何为用户级线程创建一级目录表？

答：线性地址空间的页到物理地址空间的页之间的映射用表来描述。为避免映射表占用巨大的存储器资源，所以把页映射表分为两级。

  页映射表的第一级称为页目录表，存储在一个4K字节的物理页中。页目录表共有1K个表项，其中，每个表项为4字节长，包含对应第二级表所在物理地址空间页的页码。

4、在GeekOS 系统中，如何为用户级线程创建二级页表？

答：页映射表的第二级称为页表，每张页表也安排在一个4K字节的页中。每张页表都有1K个表项，每个表项为4字节长，包含对应物理地址空间页的页码。由于页目录表和页表均由1K个表项组成，所以使用10位的索引就能指定表项，即用10位的索引值乘以4加基地址就得到了表项的物理地址。

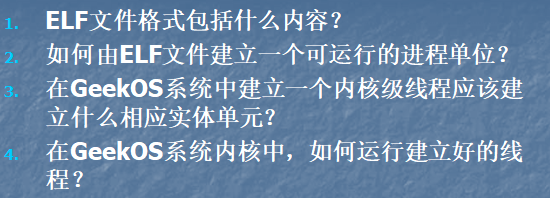
1. 在GeekOS系统中，如何分页存储管理来运行用户级线程？

答：给每个进程分配一张页表，当要运行该用户线程时，只要将要执行进程的页表调入内存，使它驻留在内存中，就可以运行用户级线程。（其他进程的页表不必驻留在内存中）

两级表的第一级表称为页目录，存储在一个4K字节的页中，页目录表共有1K个表项，每个表项为4个字节，线性地址最高的10位（22-31）用来产生第一 级表索引，由该索引得到的表项中的内容定位了二级表中的一个表的地址，即下级页表所在的内存块号。第二级表称为页表，存储在一个4K字节页中，它包含了 1K字节的表项，每个表项包含了一个页的物理地址。二级页表由线性地址的中间10位（12-21）位进行索引，定位页表表项，获得页的物理地址。页物理地 址的高20位与线性地址的低12位形成最后的物理地址。

由于4G的地址空间划分为1M个页，因此，如果用一张表来描述这种映射，那么该映射表就要有1M个表项，若每个表项占用4个字节，那么该映射表就要占用4M字节。为避免映射表占用巨大的存储器资源，所以把页映射表分为两级。

**Project1：**



3.简在GeekOS系统中建立一个内核级线程应该建立声明相应实体单元？

该相应实体单元应该包含有一个已初始化的Kernel\_Thread结构体，和一个包括参数地址、返回地址、入口地址、寄存器值等的进程栈区，且该实体单元的进程状态应为就绪态。

（一般来说，进程创建过程为：

在进程列表中增加一项，从PCB池中申请一个空间PCB，为新进程分配唯一标识符；

通过Init\_Thread函数初始化Kernel\_Thread结构体，Kernel\_Thread结构体就是GeekOS中的PCB。

为新进程的进程映像分配地址空间，以便容纳进程实体，由进程管理程序确定加载至进程地址空间中的程序；

为新进程分配各种资源；

通过Setup\_Kernel\_Thread函数初始化进程栈区，包括参数地址、返回地址、入口地址、寄存器值等

通过Make\_Runnable\_Atomic函数将该进程设置为就绪态，加入等待执行队列。

初始化PCB，如进程标识符、处理器初始状态、进程优先级等；

把新进程的状态设置为就绪态，并将其移入就绪队列；

通知操作系统某些模块，如记账程序、性能监控程序等。）