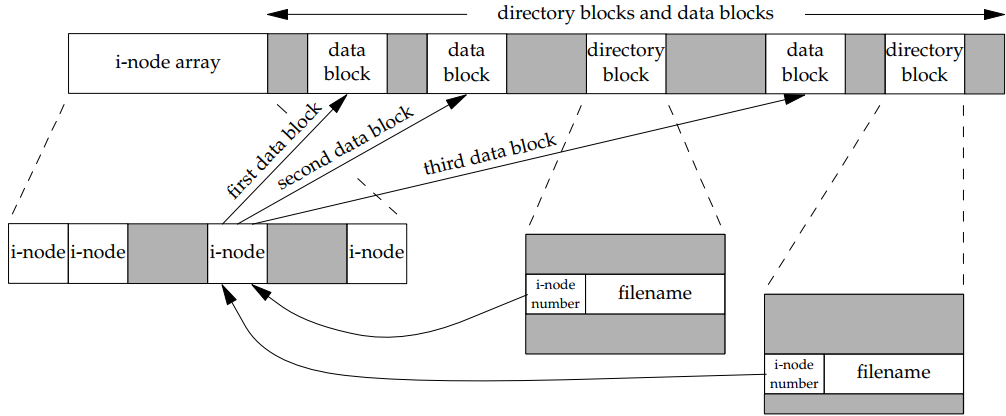
st\_size仅针对regular file，directory file以及symbolic link（文件系统主要也是，见下图）；单位是byte；

i-node不是一个点也不是一个数字而是一个结构：The i-nodes are fixed-length entries that contain most of the information about a file；



Linux文件系统组织结构概要

“文件名和文件路径名”：上图中的filename是文件名还是文件路径名？应该是文件名，文件路径名是一个查找结构（作为树级层层查找的依据），其中涉及到的目录文件是目录文件；举个例子，link函数的原型是int link(const char \*existingpath, const char \*newpath)，其中参数existingpath可以是文件路径名（相对路径或者绝对路径），但是最终存储到“i-node number: filename”中的filename还是文件名本身，不包括前面的路径，路径是被link函数用来树级层层查找最后面的目录文件（filename的直接目录），然后在目录文件中添加“i-node number: filename”目录项；

data block内容是数据，对于regular file，其数据是各种文件类型的数据，对于directory file其数据是目录项数据信息（i-node number：filename），对于symbolic link，其数据是文件名全路径信息；

可以用不同的filename（相当于同一文件的各种别名）对应同一个i-node number从而创建多个directory entry（Hard Link），只用当所有的这些别名（directory entry）被删除后（ulink），文件的data所占block才真正被释放；这些别名（Hard Link）不是一个文件，而symbolic link是一个文件（有i-node和data block），只不过它的data block是其link的文件的文件名，删除symbolic link（它自己作为文件本身也有hard link）只会删除它的data block，不会删除其指向文件的directory entry；

link类似于同一磁盘下（同file system）的复制，在各个文件夹下都有复制品，但真正的文件数据只有一个，只有将每个文件的文件删除（ulink），真正的文件数据才会被释放（“这个比喻不对，因为它们的i-node number一样，所以共享同一个i-node，所以共享数据块，所以在一个地方修改文件，另一个地方也会受到影响”）

rename和renameat函数是把同一个directory下的entry的filename改变还是可以连文件路径名也可以（如果是后者，那么源directory file的entry消失，新directory file的entry建立）？？？

rename和renameat函数作用原理：If newname already exists, we need permissions as if we were deleting it. Also, because we’re removing the directory entry for oldname and possibly creating a directory entry for newname, we need write permission and execute permission in the directory containing oldname and in the directory（不是复数形式，所以都是直接目录，因为是修改直接目录的entry） containing newname