1、 交换技术用于多道程序环境，可以提高各作业的响应时间。

所谓交换，就是系统根据需要把内存中暂时不运行的某个(或某些)作业部分或全部移到外存，而把外存中的某个(或某些)作业移到相应的内存区，并使其投入运行。

交换的时机通常在以下情况发生：

——作业的进程用完时间片或等待输入输出。

——作业要求扩充存储而得不到满足时。

实现：

通常把辅存分为文件区和交换区，文件区用于存放文件，交换区用于存放从内存中换出的作业（进程）。在交换区的作业驻留时间是短暂的，交换操作又较为频繁，所以对交换区管理的主要目的是提高作业的换入换出速度，故对交换区采用连续分配方式。交换技术的关键是设法减少每次交换的信息量。为此，常将作业的副本保留在外存，每次换出时，仅换出那些修改过的信息即可。

2、指令和数据绑定到存储器地址可在下面任意时期进行：

1.编译时

如果在编译时就知道进程将在内存中的驻留地址，那么就可以生成绝对代码。绝对代码：这段代码被编译成在一个在特定的地址工作的代码，并且只在加载到那个特定的地址时才工作。分支和跳转指令都包含一个固定的精确（绝对）地址。它是一种通常在嵌入式系统中找到的代码类型，因此可以保证一段代码将加载到该特定地址，因为它是唯一加载到该地址的代码。

2.加载时

如果在编译时并不知道进程将驻留在内存的什么地方，那么编译器就必须生成可重定位代码(relocatable code)。对于这种情况最后绑定会延迟到加载时才进行。如果开始地址发生变化，只需要重新加载用户代码以引入改变值。

3.执行时

如果进程在执行时可以从一个内存段移到另一个内存段，那么绑定必须延迟到执行时才进行。采用这种方案需要特定的硬件。绝大多数通用计算机操作系统采用这种方法。

3、 MMU、快表、页面错误异常、页表、页表项、地址保护、放置策略、清除策略、置换策略、预取策略、驻留集/工作集

程序要读取页面内容，通过 MMU 中的快表来获取页面的物理地址，如果对应的快表项不存在，就通过地址保护机制检查逻辑地址是否越界或者越权，如果通过检查就到内存的页表中去查找，如果存在对应的页表项就把该项拷贝到快表，程序通过读取快表得到对应的物理地址，如果对应的页表项不存在，则发生缺页异常，系统根据清除策略查看空闲链表是否有需要的页框，查看驻留集是否已满，如果满了，就根据置换策略，将相对不重要的页框丢弃，根据放置策略添加新的页框到驻留集，此时可以通过预读策略多读取一些可能之后会访问的内容。

4、 普通页表：虚拟地址空间是 2^48，即没有完全使用 64 位地址。页面大小为 2^12，即 4KB，则用于分页的地址线的位数为 48-12=36。若采用普通页表，每个页表项为 4 字节，则页表占据的内存空间为 2^38B。若采用多级页表，页表占据的内存空间要大于 2^38B。采用 hash 表的反转页表思路：为这个系统建立一张页表，考虑物理内存空间为 4GB，则用户空间的最大页数为 2^20 页，若一条页表项占内存空间为 4 字节，则页表所占内存空间最大为 2^22B。

5、 linux 的地址空间由用户部分与内核部分组成，通过上下文切换来改变当前的用户地址空间，而内核地址空间是不变的。用户模式的程序不能访问内核虚拟地址空间。内核逻辑地址通过 kmalloc 分配，不能被交换，虚拟地址连续分配等于物理地址连续分配。内核

虚拟地址通过 vmalloc 分配，可以分配大于物理地址空间的地址。

6、 mmap 实际上是将磁盘文件映射到进程虚拟地址的机制，它读取与写入操作的效率高于直接使用读写系统调用，因为它在实现上少了将文件拷贝到用户地址空间与从用户地址空间拷贝的步骤，mmap 是实现读操作是将磁盘文件拷贝到内核地址空间，程序可以直接访问该空间，而 read 系统调用是将磁盘文件拷贝到内核地址空间，然后再拷贝到程序的地址空间，程序访问自己的地址空间，写操作也是类似的，可是 mmap 的写操作若 mmap 地址未对应物理内存，则产生缺页异常，所以当处理大数据时，write 系统调用可能会比 mmap 写操作好，所以在读数据时使用 mmap 最好，写数据时可以考虑 mmap。mmap 有两种类型，一种是有 backend，一种是没有 backend，mmap 的模式参数为 MAP_SHARED 与 MAP_PRIVATE 属于前者，MAP_NORESERVE 属于后者。MAP_SHARED 模式允许程序映射最大的地址空间，MAP_PRIVATE 因为支持 copy-on-write，副本是不会被写回的，所以这种方式与没有 backend 一样无法映射太大的地址空间。