第 8 章作业

8-1、

设备独立性是指用户在编程时使用的设备与实际使用的物理设备无关，即用户在程序中仅使用逻辑设备名，不直接使用物理设备。其中，逻辑设备名是用户自己指定的设备名，它是暂时的、可更改的。而物理设备名是系统提供的设备的标准名称，它是永久的、不可更改的。
引入设备独立性的引入是由于各设备的特性差异，为提高设备利用率，每种设备的驱动程序往往与设备物理特性相关，这给设备的使用带来了不便。
引入设备独立性概念，能够方便用户、提高设备利用率、提高系统可适应性和可扩展性。

8-5、

常用的缓冲技术有：双缓冲、环形缓冲和缓冲池。
双缓冲：为输入或输出分配两个缓冲区。提高设备的并行操作能力。例如当用于设备输入时，输入设备首先填满一个 buf，假设为 buf1；当用户进程从 buf1 提取数据时，输入设备继续填充另一个 buf，假设为 buf2。当 buf1 为空后，而 buf2 满时，用户进程又可以从 buf2 中提取数据，与此同时，输入设备又可以填充 buf1. 两个缓冲区交替使用，使 CPU 和输入设备并行操作程度进一步提高。
环形缓冲：在主存中分配一些大小相同的存储单元当作各缓冲区。每个缓冲区指向下一个缓冲区，最后一个缓冲区指向第一个缓冲区，形成一个缓冲环。需要两个指针进行输入输出的指示，in 指针指向环上第一个空缓冲区，out 指向第一个满的缓冲区。输入时，将数据输入到 in 指向的空缓冲区中，in 指针指向下一个空缓冲区；输出时，从 out 指向的缓冲区拿出数据，然后 out 指向下一个满缓冲区。
缓冲池：由一组大小相同的缓冲区组成。在缓冲区中各个缓冲区作为系统公共资源为进程所共享，并由系统进行统一分配和管理。既可以用输入也可以用于输出。当某个进程需要缓冲区时，提出申请，由系统将某些缓冲区分配给该进程；当使用完毕后，再将这些分配的缓冲区归还给缓冲池。达到少量缓冲区服务多进程的效果。缓冲池通常分成三个缓冲区队列：空闲缓冲区队列、输入缓冲区队列和输出缓冲区队列。

8-10、

虚拟设备技术：是在一类物理设备上模拟另一类物理设备的技术，是将独占设备转化为共享设备的技术。由于独占设备按静态方式分配，不利于提高系统效率，为了解决这一问题，提出虚拟设备技术。
虚拟设备：通过虚拟技术将一个独占设备转变为若干个逻辑设备，供多个用户进程使用，通常把用来代替独占设备的那部分外存空间称为虚拟设备。
虚拟分配：对上述的虚拟设备采用虚拟分配。当某进程需要与独占设备交换数据时，系统将与该独占设备所对应的那部分磁盘空间分配给它，并建立相应的数据结构。具体做法是：系统将欲从独占设备输入或输出的信息，先复制到辅存的存

储设备中，当进程需要从输入设备读取信息时，系统将这一请求转化为从辅存读入的请求，并将对应的数据从辅存中拿出；当进程需要输出数据时，则先将待输出信息存入辅存，在合适的时间，相应的输出设备将该数据从辅存中复制出来。这样可以大大提高输入输出效率。

8-23、

一个“延迟写”的块真正写到磁盘上的步骤：

“延迟写”标志指示核心在某缓冲区重新分配出去之前必须把缓冲区的内容写到磁盘上。高速缓冲模块中，当需要缓冲区分配时，如当要从一个特定磁盘块上读数据或把数据写到一个特定磁盘块上时，需要检查该块是否在高速缓冲中，若不在，则需要分配一个缓冲区给磁盘块。当该缓冲区有“延迟写”标记时，将该“延迟写”的缓冲区写到磁盘上。核心开始一个往磁盘的异步写，并且试图从空闲缓冲区队列中分配另一个缓冲区。当异步写完成后，该“延迟写”才真正写到了磁盘上。之后核心把该“延迟写”的块标记为“旧”，释放该缓冲区置于空闲缓冲区队列头部，完成整个步骤。

8-27、

（1）

```
main(){
    int empty1 = 1, empty2 = 1; //指示buf1与buf2是否为空
    int full1 = 0, full2 = 0; //指示buf1与buf2是否满
    int mutex1 = 1, mutex2 = 1; //对两个公共buf的访问进行互斥
    cobegin
        P(); D();
    coend
}
```

```
P(){
while(1){
    if(empty1){  //当buf1空时，将数据输出到buf1
        ...
        P(empty1);
        P(mutex1);
        向buf1输出数据;
        V(mutex1);
        V(full1);
        ...
    }
    else if(empty2){  //当buf2空时，将数据输出到buf2
        ...
        P(empty2);
        P(mutex2);
        向buf2输出数据;
        V(mutex2);
        V(full2);
        ...
    }
}
}
```

```
D(){
while(1){
    if(full1){  //当buf1满时，将buf1数据输出到设备D
        ...
        P(full1);
        P(mutex1);
        从buf1取出数据;
        V(mutex1);
        V(empty1);
        ...
    }
    if(full2){  //当buf2满时，将buf2数据输出到设备D
        ...
        P(full2);
        P(mutex2);
        从buf2取出数据;
        V(mutex2);
        V(empty2);
        ...
    }
}
}
```

（2）

双缓冲系统输出结果：

P	1	2	1	2	1	2	1	2	…	1	2	
D		1	2	1	2	1	2	1	2	…	1	2

图中的一个表格表示进行的一个计算结果输出，表格中的内容表示此时使用的 buf。

若有 n 个数据输出。

若 P 的计算速度 x >= D 的输出速度 y，则总用时为 1/x + n/y。

若 P 的计算速度 x < D 的输出速度 y，则总用时为 n/x + 1/y。

则双缓冲系统输出结果速度为 $\frac{n}{\frac{1}{\max(x,y)}+\frac{n}{\min(x,y)}}$

单缓冲系统输出结果：

P	1		1		1		1		…	1	
D		1		1		1		1		…	1

若有 n 个数据输出。

则总用时为 n/x + n/y。

则单缓冲系统输出结果速度为 $\frac{n}{\frac{n}{x}+\frac{n}{y}}=\frac{1}{\frac{1}{x}+\frac{1}{y}}$。

由此可见，输出 n 个结果，单缓冲系统用时比双缓冲系统用时多出(n-1)/max(x,y)时间，效率不高。