



### 3、系统调用

```
#include <fcntl.h>
char buffer[2048];
int version = 1;
```

.....

```
copyOperation (old,new)
    int old,new;
{
    int count;
    N:  while ((count=read(old,buffer,sizeof(buffer)))>0)
        write(new, buffer, count);
}
```

用户态

核心态

进程返回用户态，执行应用程序处理**buffer**数组保存的文件数据

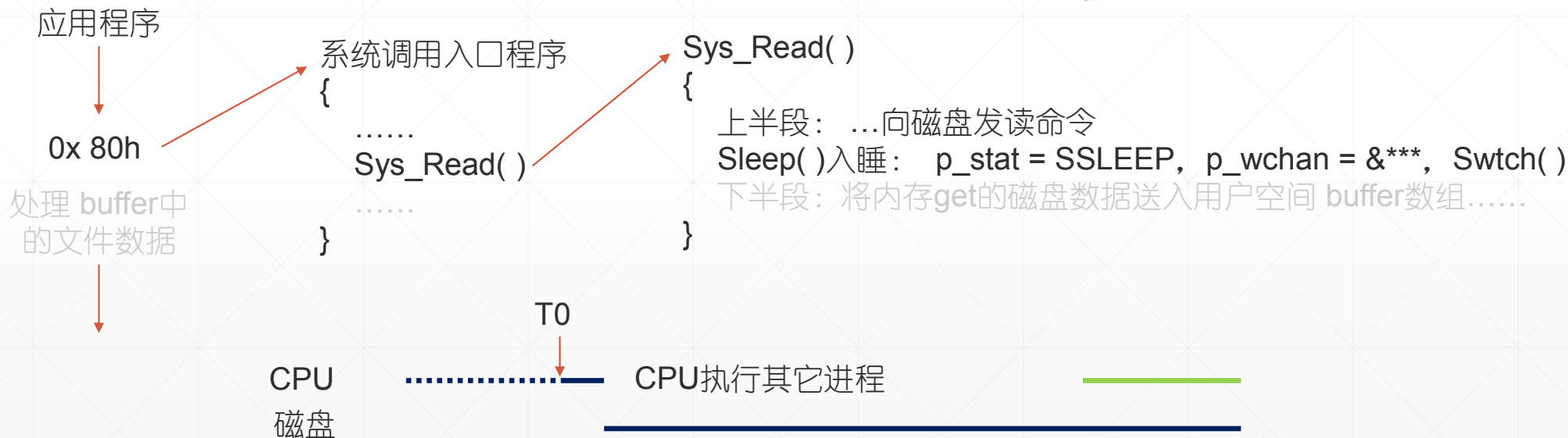
进程执行**read**系统调用，  
读磁盘文件。  
将文件数据送**buffer**数组

应用程序执行 **read** 系统调用。负责执行这个程序的进程陷入内核，读取磁盘文件数据，将其送入用户空间，**buffer**数组。完成后，系统调用结束。进程返回用户态，执行应用程序处理 **buffer**数组中的文件数据，把它写进另一个文件 **new**。

# read 系统调用的执行细节 1

T0 时刻，PA 执行应用程序，子程序 CopyOperation 执行 read 系统调用读 old 文件数据。

```
copyOperation (old,new)
int old,new;
{
    int count;
    N: while ((count=read(old,buffer,sizeof(buffer)))>0)
        write(new, buffer, count);
}
```



# read 系统调用的执行细节 2

T1时刻，磁盘IO完成。现运行进程PB用户态运行，执行showStack( )函数。



read系统调用细节:

- ① 将read系统调用号3送入EAX, 两个参数(文件名old和buffer)分别送入寄存器EBX和ECX
- ② int 0x80软中断 → CPU响应中断, 切换状态为核心态, 执行系统调用入口程序, 完成现场的保护
- ③ 执行系统调用处理程序, 向磁盘发送读命令, 调用sleep入睡, switch切换当前进程下台
- ④ 磁盘读入完毕, PA被唤醒, PA欲执行read系统调用后半段, 且优先级高, RunRun++, Switch选中PA上台执行, PA将数据写入缓冲区
- ⑤ 返回用户态, read返回值由EAX寄存器带回

write系统调用细节:

- ① write系统调用号4送入EAX寄存器, 两个参数(new, buffer)送入EBX和ECX寄存器
- ② int 0x80软中断 → CPU响应中断, 切换至核心态, 执行系统调用入口程序, 保护现场
- ③ 执行系统调用执行程序, 向磁盘发送写命令, 调用sleep入睡, switch切换当前进程下台
- ④ 磁盘写入完毕, PA被唤醒, PA欲执行write后半段, PA的优先级高, RunRun++, PA被Switch选中, 完成write系统调用后半段
- ⑤ 返回用户态

read调用的执行细节可分为CPU侧和磁盘硬件侧

① CPU侧:  $T_0$ 时刻, PA进程执行read系统调用陷入内核, 执行系统调用上半段, 向磁盘发出IO命令

② 磁盘硬件侧:  $[T_0, T_1]$ 时间段执行IO操作, 读取文件数据, 完成后向

## CPU发中断请求

- ③ CPU侧:  $T_1$ 时刻。响应磁盘中断。现运行进程PB进入内核执行磁盘处理中断程序: 读取从磁盘里送来的数据, 存入核心卷内存, 唤醒睡眠进程PA。完成后PB中断返回, 例行调度, PB被剥夺, 执行switch()放弃CPU。PA作为优先级最高的进程被选中, 成为新运行进程
- ④ CPU侧: PA进程sleep返回, 执行read系统调用下半段, 将核心卷缓存中存放的文件数据送入用户空间buffer数组

write与read大致类似, 只是中断处理程序不同, 向磁盘发写命令。