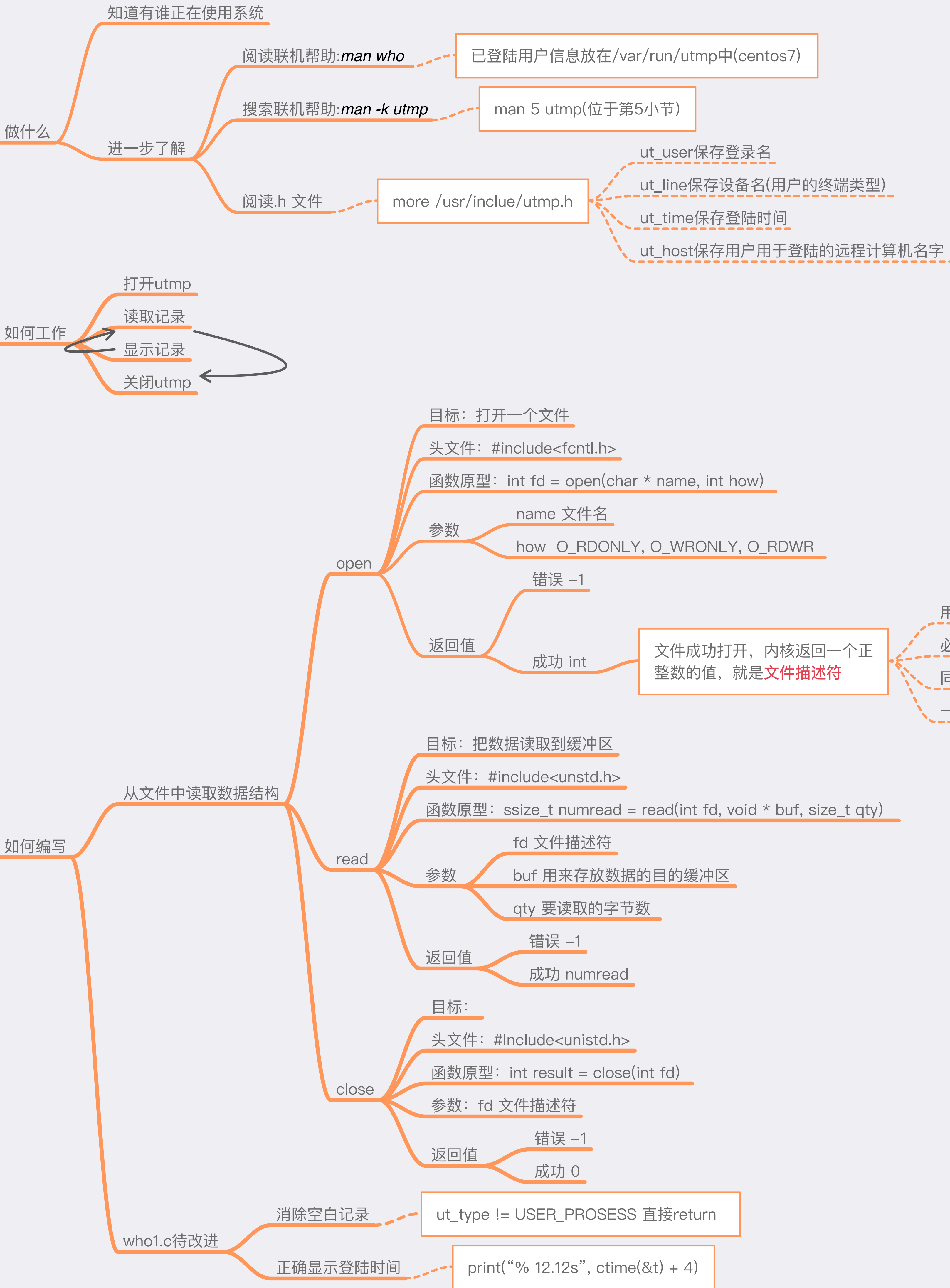


第2章 用户、文件操作与联机帮助

who命令



cp命令



使用缓冲提高文件I/O效率

缓冲区的大小对性能有很大的影响, 原因是系统调用需要在用户-内核之间转换, 会消耗很多时间

原来的 `who` 命令每次只取一个用户, 大大浪费了时间, 可以一次取16条记录

内核缓冲技术

内核也使用缓冲技术来提高对磁盘的访问速度, 磁盘是数据块的集合, 内核会对磁盘上的数据块做缓冲

内核将磁盘的数据块复制到内核缓冲区中, 当一个用户空间中的进程要从磁盘上读数据时, 内核一般不直接读磁盘, 而是将内核缓冲区中的数据复制到进程的缓冲区中

当进程所要求的数据块不在内核缓冲区时, 内核会把相应的数据块加入到请求数据列表中, 然后把该进程挂起, 接着为其他进程服务

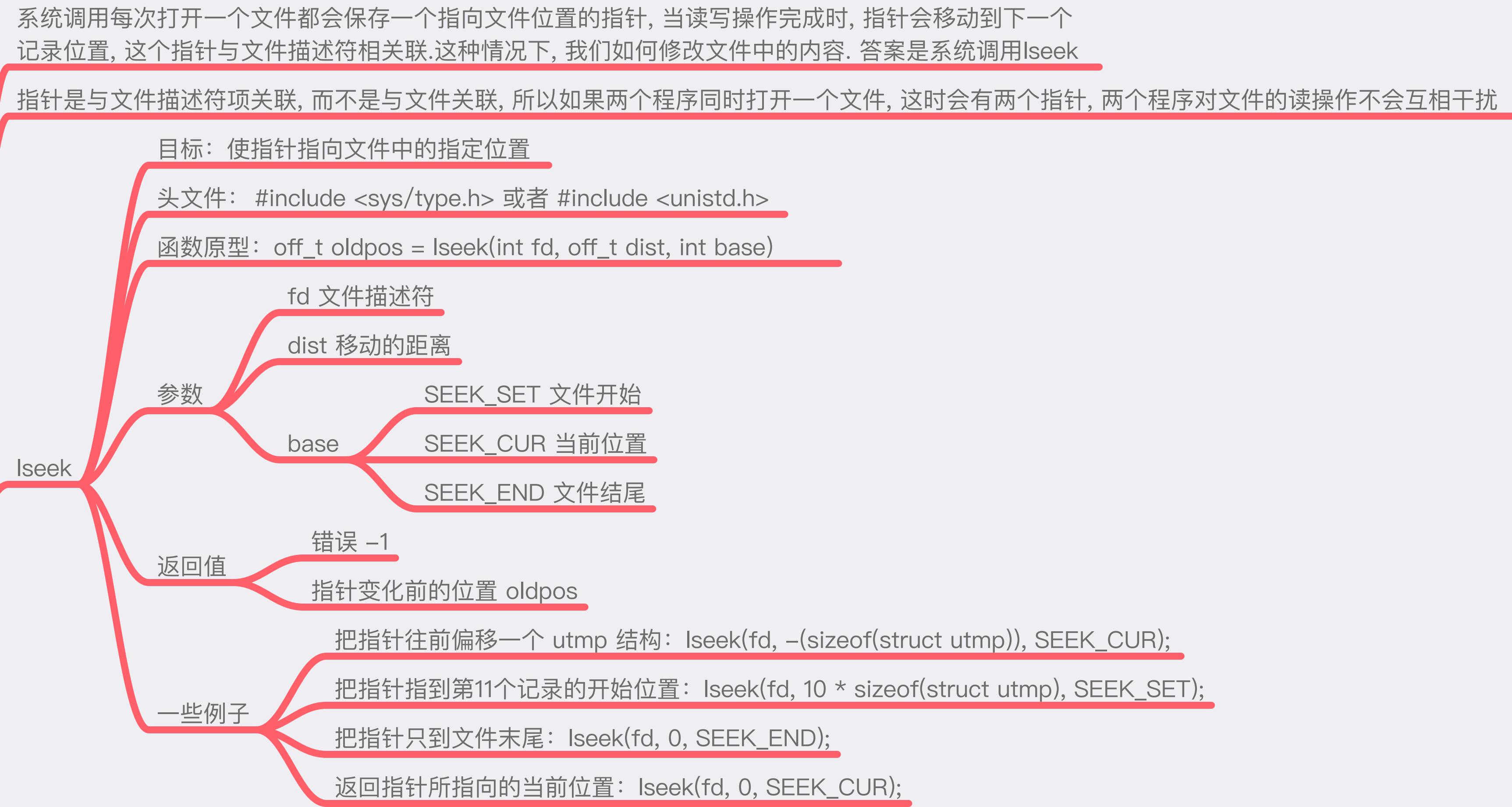
当内核把相应的数据读到内核缓冲区, 然后再把数据复制到进程缓冲区中, 最后唤醒被挂起的进程

提高磁盘I/O效率

优化的结果

- 优化磁盘的写操作
- 需要及时地将缓冲数据写入磁盘

文件读写



终端注销代码

```
logout_tty(char *line)
{
    int fd;
    struct utmp rec;
    int len = sizeof(struct utmp);
    int retval = -1;

    if ((fd = open(UTMP_FILE, O_RDWR)) == -1)
        return -1;

    while (read(fd, &rec, len) == len)
    {
        if (strcmp(rec.ut_line, line, sizeof(rec.ut_line)) == 0)
        {
            rec.ut_type = DEAD_PROCESS;
            if (time(&rec.ut_time) != 0)
            {
                if (lseek(fd, -len, SEEK_CUR) != -1)
                {
                    if (write(fd, &rec, len) == len)
                        retval = 0;
                    break;
                }
            }
        }
        if (close(fd) == -1)
            retval = -1;
        return retval;
    }
}
```

上面函数接受一个 `line` 参数, 用于对比 `struct utmp` 中的 `ut_line` 成员, 来确定当前记录, 读到当前记录后, 更改属性, 并使用 `lseek` 将文件指针回退一个 `utmp` 长度, 并将新的 `utmp` 写入到文件

处理系统调用中的错误

