操作系统实验手册：内核 API

1. 实验题目

1.1 实验目的

通过本实验的学习，掌握信创操作系统内核定制中所常用的内核数据结构和函数，具体包括：i.内核链表；ii. 内核内存的分配释放； iii.内核线程；iv.内核锁；

1.2 实验内容

设计一个内核模块，并在此内核模块中创建一个内核链表以及两个内核线程。

- 线程1需要遍历进程链表并将各个进程的 pid、进程名加入到内核链表中。
- 线程2中需不断从内核链表中取出节点并打印该节点的元素。

在卸载模块时停止内核线程并释放资源。

2. 实验结果

使用 dmesg 命令可以观察到当前系统中运行的所有进程的 pid 与进程名。

```
[  560.790064] pid :1204        name:crond
[  560.790064] pid :1216        name:agetty
[  560.790065] pid :1389        name:dhclient
[  560.790065] pid :1506        name:rsyslogd
[  560.790065] pid :1846        name:sshd
[  560.790066] pid :1850        name:sshd
[  560.790066] pid :1851        name:sh
[  560.790066] pid :1923        name:code-4849ca9bdf
[  560.790067] pid :1951        name:sh
[  560.790067] pid :1955        name:node
[  560.790068] pid :1979        name:node
[  560.790068] pid :2040        name:node
[  560.790069] pid :2046        name:node
[  560.790069] pid :2091        name:sh
[  560.790069] pid :2241        name:cpptools
[  560.790070] pid :2247        name:node
[  560.790070] pid :2272        name:bash
[  560.790070] pid :2402        name:sleep
[  560.790071] pid :2425        name:make
[  560.790071] pid :2655        name:kworker/u256:0
[  560.790072] pid :2801        name:sudo
[  560.790072] pid :2812        name:cpptools-srv
[  560.790072] pid :2821        name:cpuUsage.sh
[  560.790073] pid :2826        name:sleep
[  560.790073] pid :2827        name:insmod
[  560.790074] pid :2828        name:thread1
[  560.790074] pid :2829        name:kthreadd
```

3. 实验环境及平台

- 操作系统：openKylin
- 内核版本：Linux 5.10.0
- 处理器数量：2
- 内存：4GB

4. 实验前置要求

本实验需要了解 kmalloc、kfree、内核链表、锁、内核线程、同步互斥相关知识。

5. 内核 API 介绍

5.1 内核内存管理

5.1.1 kmalloc

void *kmalloc(size_t size, gfp_t flags);

说明：

- kmalloc 用于在内核空间分配内存。

参数：

- size：请求分配的内存大小。
- flags：分配标志，用于指定内存分配的行为和属性
  - GFP_KERNEL：普通的内核内存分配，可能会休眠。
  - GFP_ATOMIC：原子内存分配，不会休眠，常用于中断上下文。

返回值：

- 被分配的内存地址。

5.1.2 kfree

void kfree(const void *objp);

说明：

- kfree 用于释放由 kmalloc 分配的内存。

参数：

- objp：指向要释放的内存块的指针。

返回值：无

5.2 内核链表

linux 内核中实现了链表结构以方便开发人员使用，被称为内核链表。内核链表为循环链表模式，传统链表包含数据域和指针域，内核链表有别于传统链表就在节点本身不包含数据域，只包含指针域。故而可以很灵活的拓展数据结构，应用更方便灵活。内核链表部分常用接口如下所示：

5.2.1 链表初始化

宏 INIT_LIST_HEAD(struct list_head *list);

说明：

- 此宏用于初始化一个链表头。

参数：

- list：带初始化链表头

返回值：无。

5.2.2 添加到链表头部

void list_add(struct list_head *new, struct list_head *head);

说明：

- 将新节点添加到链表的头部

参数：

- new：待添加节点。
- head：原链表头。

返回值：无。

5.2.3 添加到链表尾部

void list_add_tail(struct list_head *new, struct list_head *head);

说明：

- 将新节点添加到链表的尾部。

参数：

- new：待添加节点。
- head：原链表头。

返回值：无。

5.2.4 删除节点

void list_del(struct list_head *entry);

说明：

- 从链表中删除指定节点。

参数：

- entry：待删除节点。

返回值：无。

5.2.5 检查链表是否为空

int list_empty(const struct list_head *head);

说明：

- 检查链表是否为空，如果为空则返回非0值。

参数：

- head：链表头。

返回值：

- 1：空
- 0：非空

5.2.6 获取第一个节点

宏 list_first_entry(ptr, type, member);

说明：

- 通过 list_head 指针 ptr，获取该链表第一个节点

参数：

- ptr：链表头
- type：数据节点类型
- member：在结构体内部的 list_struct 名称

返回值：

- 链表第一个节点

5.2.7 遍历链表

宏 list_for_each_safe(pos, n, head)

说明：

- 这个宏可以在遍历时删除节点。

参数：

- pos：当前 list_head 结构体
- n：下一个 list_head 结构体
- head：链表头

返回值：无。

5.3 内核线程

内核线程是直接由内核本身启动的进程。内核线程实际上是将内核函数委托给独立的进程，它与内核中的其他进程”并行”执行。内核线程经常被称之为内核守护进程 。

5.3.1 创建内核线程

宏 kthread_create(threadfn, data, namefmt, arg...)

说明：

- 创建一个内核线程，并返回其句柄。（创建好的线程不会直接运行，需要调用 wake_up_process）

参数：

- threadfn：线程执行体。需要是一个返回值为 int，参数为 void*的函数。

- data：传入到线程执行体中的数据。
- namefmt：线程名。

返回值：

- 内核线程句柄。

5.3.2 唤醒休眠线程

int wake_up_process(struct task_struct *tsk)

说明：

- 用于唤醒处于睡眠状态的进程，使进程由睡眠状态变为 RUNNING 状态，从而能够被 CPU 重新调度执行。

参数：

- tsk：内核线程句柄

返回值：

- 1：唤醒成功
- 0：唤醒失败

5.3.3 内核线程的退出

int kthread_stop(struct task_struct *thread);

说明：

- 向目标内核线程发送退出信号，等待其退出。

参数：

- thread：内核线程句柄

返回值：

- 返回线程执行体的结果，如果从未调用 wake_up_process()，则返回 -EINTR。

5.4 自旋锁

5.4.1 自旋锁初始化

宏 spin_lock_init(_lock)

说明：

- 初始化自旋锁

参数：

- _lock：自旋锁句柄

返回值：无。

5.4.2 加锁

void spin_lock(spinlock_t *lock)

说明：

- 自旋锁加锁

参数：

- lock：自旋锁句柄

返回值：无

5.4.3 解锁

void spin_unlock(spinlock_t *lock)

说明：

- 自旋锁解锁

参数：

- lock：自旋锁句柄

返回值：无

5.5 内核休眠

unsigned long msleep_interruptible(unsigned int msecs)

说明：

- 可以被信号唤醒的休眠。

参数：

- msecs：休眠时长，单位 ms

返回值：

- 休眠的时长。

5.6 进程链表

5.6.1 遍历进程链表

宏 for_each_process(p)

说明：

- 遍历系统中所有进程。

参数：

- p：会在遍历过程中指向每一个进程的 task_struct 结构体。

返回值：无