

# 实验案例四：内核子系统—进程管理(指导文档)

## 实验案例四：内核子系统—进程管理(指导文档)

- 一、实验简介
- 二、实验内容步骤
  - 1.进程信息打印

### 一、实验简介

在进行 OpenHarmony 进程管理实验之前，需要理解进程并非所有操作系统内核通用的概念。OpenHarmony 当前支持的三种内核中，只有 **Linux** 和 **LiteOS-A** 提供进程机制；而 **LiteOS-M** 内核由于缺乏 MMU，无法实现不同进程间的地址空间隔离，仅提供任务（tasks）机制。

本实验基于 LiteOS-A 内核，旨在学习和理解其进程管理机制。实验过程中需深入阅读 LiteOS-A 源码，并在 OpenHarmony 内核子系统中进行编程，实现与进程管理相关的功能，从而加深对 LiteOS-A 内核设计与实现的理解。

### 二、实验内容步骤

本次实验需要依次完成下列实验任务，各个步骤详情如下。

#### 1.进程信息打印

本实验基于实验案例三的系统调用实验。在此基础上，需要修改 LiteOS-A 内核中处理系统调用中断的函数实现，使得在调用新创建的系统调用时，能够返回当前系统中所有正在运行的进程信息，并打印至控制台。实验预期效果如下：

```
OHOS:/$ ./bin/print_taskinfo
User mode:: SYS_print_taskinfo
Kernel mode: SYS_print_taskinfo
```

PID	PPID	PGID	Mode	Status	Priority	PName
0	2	2	kernel	Pending	31	KIdle
1	-1	1	user	Pending	28	init
2	-1	2	kernel	Pending	0	KProcess
3	1	3	user	Pending	28	mksh
5	3	5	user	Pending	28	print_taskinfo

#### 流程

由于本节任务一的程序需要打印所有进程信息，该功能需在内核态实现，因此通过系统调用完成。在实验案例三中添加系统调用的基础上，本节实验需修改原程序中对应的系统调用函数，使其能够遍历系统中所有进程并打印相关信息。具体而言，只需修改 `kernel/liteos_a/syscall/my_syscall.c` 文件，在其中实现打印进程信息的功能。

OpenHarmony LiteOS 内核的进程数量受全局变量 `g_processMaxNum` 限制，且进程号在 `[0, g_processMaxNum]` 范围内。遍历系统中所有进程时，可通过宏 `OS_PCB_FROM_RPID()` 获取该范围内的进程控制块（PCB），并判断其状态是否被使用。对于状态标志位不为 `OS_PROCESS_FLAG_UNUSED` 的 PCB，依次打印其相关信息。具体实现代码如下：

```

void SysPrintTaskinfo(void) {
    LosProcessCB* processCB;
    PRINTK("kernel mode: SYS_print_taskinfo\n");
    PRINTK("\r\n PID PPID PGID Mode Status Priority PName\n");
    for(int index=0; index < g_processMaxNum; index++) { // 遍历所有进程号
        processCB = OS_PCB_FROM_RPID(index); // 由进程号取得PCB
        if(processCB->processStatus & OS_PROCESS_FLAG_UNUSED) continue; // 查询PCB的标志位

        PRINTK(" %4u", processCB->processID); // 打印进程ID
        if(processCB->parentProcess) // 查询并打印父进程ID
            PRINTK("%6u", processCB->parentProcess->processID);
        else
            PRINTK("%6d", -1);
        PRINTK("%5u", OS_GET_PGROUP_LEADER(processCB->pgroup)->processID); // 打印进程所属组
ID
        PRINTK("%9s%8s%9u", ConvertProcessModeToString(processCB->processMode),
ConvertProcessStatusToString(processCB->processStatus), GetProcessPriority(processCB)); //打印进程状态
        PRINTK(" %-32s\n", processCB->processName); // 打印进程名
    }
    return;
}

```

其它相关代码如下：

```

STATIC UINT8 *ConvertProcessModeToString(UINT16 mode)
{
    if (mode == OS_KERNEL_MODE) {
        return (UINT8 *)"kernel";
    } else if (mode == OS_USER_MODE) {
        return (UINT8 *)"user";
    }

    return (UINT8 *)"ERROR";
}

STATIC UINT8 *ConvertProcessStatusToString(UINT16 status)
{
    if (status & OS_PROCESS_STATUS_ZOMBIES) {
        return (UINT8 *)"Zombies";
    } else if (status & OS_PROCESS_STATUS_INIT) {
        return (UINT8 *)"Init";
    } else if (status & OS_PROCESS_STATUS_RUNNING) {
        return (UINT8 *)"Running";
    } else if (status & OS_PROCESS_STATUS_READY) {
        return (UINT8 *)"Ready";
    } else if (status & OS_PROCESS_STATUS_INACTIVE) {
        return (UINT8 *)"Inactive";
    }

    return (UINT8 *)"Pending";
}

```

```
UINT16 GetProcessPriority(LosProcessCB* processCB) {
    LosTaskCB* taskGroup = processCB->threadGroup;
    SchedParam param = {0};
    taskGroup->ops->schedParamGet(taskGroup, &param);
    switch (param.policy)
    {
        case LOS_SCHED_DEADLINE:
            return 32;
            break;
        default:
            return param.basePrio;
            break;
    }
}
```

完成修改后，重新编译并运行 OpenHarmony，然后在 OHOS 中执行程序 `./bin/print_taskinfo` 即可。