

非抢占式内核

任务 1: 多 tasks 启动与 context switch

冯吕，张旭

University of Chinese Academy of Sciences

2017 年 10 月 11 日



中国科学院大学

Outline

① 任务

② 实现

③ 问题及解决方法

整个 task 的实现。

[illegible]

函数以及函数调用关系

- *bootblock*读盘 \Rightarrow *_stat()*
_stat() 函数完成 *PCB* 的初始化和准备队列的初始化, 之后调用 *scheduler_entry()* 启动进程 (线程 1);
- *scheduler_entry()* \Rightarrow *scheduler()* \Rightarrow 启动第一个进程
- 线程切换: *do_yield()* \Rightarrow
save_PCB() && *queue_push_scheduler_entry()*
- 进程切换: *yield()* \Rightarrow 系统调用 \Rightarrow *do_yield()*

需要实现下面的函数：

- 修改 `createimage`，需要将 `process1,2,3` 也写入 `image`.
- *PCB* 的设置
- 栈的设置
- `_stat()`：初始化 *PCB* 和准备队列。
- `save_pcb()`：保存上下文
- `scheduler_entry()`：选取下一个进程来运行
- ...

PCB 结构

需要保存的信息：

- $s0 \sim s7$ ：寄存器变量，切换时需要保存。
- sp ：栈指针
- ra ：返回地址
- 进程状态（在任务一中暂时没什么用）。

栈的设置

不能过大，否则 \Rightarrow TLB miss.
不能太小，否则与进程地址冲突。

函数实现

看代码。

入口地址变化

- 在实验一中，读完后调到 `0xa080026c` 处执行。
- 在实验二中，`_stat()` 函数的入口地址变为 `0xa08002bc`。

PCB 信息存储

- 队列结构中只有一个 PCB 指针的指针，kernel.c 中有一个 PCB 指针数组，所以还需要自己声明一个 PCB 数组存储信息。
- $queue \rightarrow pcbs \Rightarrow ready_array[] \Rightarrow$ 真正存储 PCB 信息的数组

栈的设置

- 栈的设置过大，导致跳到线程 1 压栈时出现 TLB miss.
- 减小栈的地址。
- 貌似一开始没有什么好的方法来确定，只能通过出错来摸索。

保存 PCB 出错

保存 PCB 的时候 sp 和 ra 会发生变化。

```
a0800660 <do_yield>:  
a0800660: 27bdf fe8      addiu    sp,sp,-24  
a0800664: afbf 0010      sw      ra,16(sp)  
a0800668: 0c20 0159      jal     a0800564 <save_pcb>  
a080066c: 0000 0000      nop  
a0800670: 3c05 a080      lui     a1,0xa080  
a0800674: 8ca3 2098      lw      v1,8344(a1)  
a0800678: 2402 0001      li      v0,1
```

kernel 入口地址

- 刚开始忘记修改 kernel 入口地址，导致进入 process1 后无法切换到内核态。