Project 3 Interactive OS and Process Management 设计文档

中国科学院大学

[姓名] 李国峰

[日期] 2021 年 1 月 25 日

1. Shell 设计（非必须项）

实质上就是一个字符串的输入和解析程序，根据输入的指令信息进行相关函数的调用即可。

2. kill 和 wait 内核实现的设计

（1） kill 处理过程中如何处理锁，是否有处理同步原语，如果有处理，请说明。

在 pcb 中设置一个数组用于存储该进程持有的锁的信息，同时设置一个变量存储锁的个数，在调用 kill()时遍历该数组，释放该数组中的每一个锁。

（2） wait 实现时，等待的进程的 PCB 用什么结构保存？

在被等待的进程中设置一个等待的进程队列，任何进程调用 wait()时都将该进程放入这个被等待进程的等待队列中。

```
int do_waitpid(pid_t pid)
{
    pcb_t *tmp;
    tmp = find_pcb(pid);
    if(tmp != NULL && tmp != current_running)
        do_block(&tmp->wait_queue);

}
```

（3） 设计或实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需项）

在实现关于栈的部分时没有考虑到包括 shell 在内的所有测试进程均为用户态进程，因此误用了命名为 kernel_stack 的栈来存储用户态进程的信息，经过一番修改后反而引起了新的 bug，为了节省时间保留了这一不足的部分。

3. 关键函数功能

（1） do_spawn()

与前一实验的 set_pcb()不同的是，需要考虑 pcb 的退出或死亡引出的存储空间的分配和回收，因此需要建立特定的数据结构来保存这些信息。本实验中用了一些全局队列，用于保存进程消亡释放的 pcb 和栈空间。当有新的进程启动时，先去这些全局队列中寻找空间，如果没有空闲空间再申请新的空间。完成 pcb 的初始化后，再将它放入就绪队列中等待调度。

```
int do_spawn(task_info_t *task)
{
    pcb_t *new_pcb;
    int i;

    if(queue_is_empty(&exit_pcb_queue))
        new_pcb = &pcb[pcb_array_p++];
    else
        new_pcb = queue_dequeue(&exit_pcb_queue);

    //basic info
    new_pcb->pid = process_id++;
    new_pcb->status = TASK_READY;
    new_pcb->type = task->type;
    new_pcb->lock_top = 0;
    for(i = 0; task->name[i] != '\0'; i++)
        new_pcb->name[i] = task->name[i];
    new_pcb->name[i] = '\0';

    queue_init(&new_pcb->wait_queue);

    //stack
    if(queue_is_empty(&exit_kernel_stack_queue))
    {
        stack_base += 0x10000;
        stack_top -= 0x10000;
        //$29 sp
        new_pcb->kernel_stack_top = stack_base;
        new_pcb->kernel_context.regs[29] = stack_base;
        new_pcb->user_stack_top = stack_top;
        new_pcb->user_context.regs[29] = stack_top;
    }
    else
    {
        stack_t *kernel_sp = queue_dequeue(&exit_kernel_stack_queue);
        //stack_t *user_sp   = queue_dequeue(&exit_user_stack_queue);
        new_pcb->kernel_stack_top = kernel_sp->stack_top;
        new_pcb->kernel_context.regs[29] = kernel_sp->stack_top;
        //new_pcb->user_stack_top = user_sp->stack_base;
        //p new_pcb->user_context.regs[29] = user_sp->stack_base;
    }
```

参考文献

[1] 中国科学院大学操作系统实验手册 Project3-MIPS（2020 秋季学期）