

# OS LAB 3 进程切换

## 个人信息

---

姓名：郑凯琳

学号：205220025

邮箱：[205220025@smail.nju.edu.cn](mailto:205220025@smail.nju.edu.cn)

## 1. 实验目的

---

完成自制简单操作系统的进程管理功能，通过实现一个简单的任务调度，介绍**基于时间中断进行进程切换**完成任务调度的全过程，主要涉及到 `fork`、`sleep`、`exit` 等库函数和对应的处理例程实现。

- (1) 内核：实现进程切换机制，并提供系统调用 `fork`、`sleep`、`exit`
- (2) 库：对上述系统调用进行封装
- (3) 用户：对上述系统调用进行测试

## 2. 实验进度

---

成功完成了所有内容。

## 3. 实验过程

---

### (1) 完成库函数

代码修改位置：lab3/lib/syscall.c

库函数可以看作系统调用的前一等级的封装。

实际上，库函数是在调用系统调用即 `irq_Handle`，只需要设置好参数传递的顺序，接下来的工作交给系统调用。

### (2) 时钟中断处理

代码修改位置：kernel/kernel/irqHandle.c

**timerHandle**

函数执行过程：

1. 遍历所有 PCB 进程，检查阻塞的进程是否需要减少睡眠时间并唤醒。
2. 检查当前进程是否需要继续运行或抢占。
3. 选择下一个可运行的进程。
4. 切换到选定进程的执行上下文，包括恢复其堆栈和寄存器状态。

函数功能：

- 处理处理器的时间中断事件。
- 对阻塞的进程进行睡眠时间的递减，并在睡眠时间为 0 时唤醒。
- 确定当前进程是否需要继续运行或抢占，若需要抢占则选择下一个可运行的进程。
- 切换到选定进程的执行上下文，使其开始执行。

timerHandle 函数实现了操作系统中的进程调度逻辑，负责在处理器时间中断发生时，选择下一个要执行的进程，并切换到该进程的执行上下文，实现了多任务操作系统中的进程切换和调度功能。

### (3) 系统调用例程

代码修改位置：kernel/kernel/irqHandle.c

对于 fork()、sleep()、exit() 三个函数，分别执行相应的系统调用。

在 syscallHandle 函数中，增加对应的 fork、sleep、exit 对应调用号的分支。

## syscallFork

函数执行过程：

- 寻找空闲 PCB：首先遍历 PCB 数组，查找一个状态为 STATE\_DEAD 的 PCB，即一个未被使用的 PCB，用于存储新进程的信息。
- 复制用户空间数据：如果找到了空闲的 PCB，启用中断，然后通过循环将当前进程的用户空间数据复制到新进程的用户空间。这里复制了 0x100000 字节（1MB）的数据，对应于用户空间的大小。
- 设置新 PCB：设置新的 PCB。它将新进程的堆栈顶部和前一个堆栈顶部设置为与当前进程对应的值，以确保新进程的堆栈与当前进程的堆栈相同。然后，将新进程的状态设置为 STATE\_RUNNABLE，并将其睡眠时间和运行时间都设置为 0。
- 设置寄存器：继续设置新进程的寄存器。设置了新进程的各个寄存器，包括通用寄存器（eax、ebx、ecx、edx 等）、指令指针（eip）、堆栈指针（esp）以及标志寄存器（eflags）。同时，也设置了新进程的段寄存器（cs、ss、ds、es、fs、gs）。
- 设置返回值：设置了新进程的 eax 寄存器为 0，表示新进程是子进程，然后设置当前进程的 eax 寄存器为新进程的 PID，以返回给父进程。
- 处理无空闲 PCB 情况：如果没有找到空闲的 PCB，函数将当前进程的 eax 寄存器设置为 -1，表示无法创建新进程。

函数功能：创建一个新的进程，并将其与当前进程的用户空间数据和寄存器状态进行复制和设置。

## syscallSleep

函数执行过程：

- 参数检查：检查传入的睡眠时间 `sf->ecx` 是否小于等于 0。如果是，则说明没有需要睡眠的时间，直接返回，不进行睡眠操作。
- 设置阻塞状态和睡眠时间：如果传入的睡眠时间大于 0，则将当前进程的状态设置为阻塞状态 (`STATE_BLOCKED`)，表示当前进程正在睡眠。同时，将当前进程的睡眠时间设置为传入的睡眠时间。
- 发出中断请求：通过发送中断请求的方式，触发系统的中断处理程序。这个中断处理程序负责在一段时间后唤醒被阻塞的进程。

函数功能：使当前进程进入睡眠状态，并在指定的时间后恢复运行。

## syscallExit

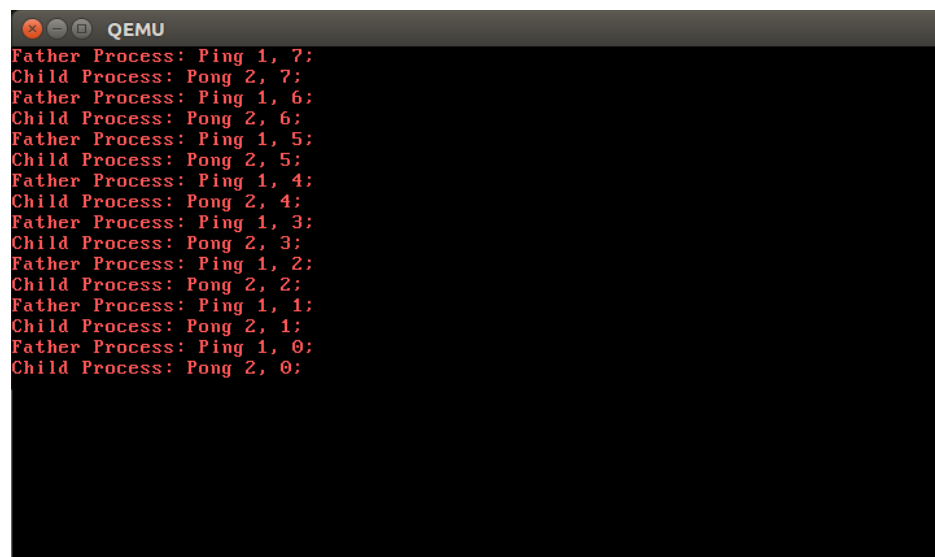
函数执行过程：

- 设置死亡状态：将当前进程的状态设置为死亡状态 (`STATE_DEAD`)。这意味着当前进程不再处于运行状态，即将从进程列表中移除。
- 发送中断请求：通过发送中断请求的方式，通知系统当前进程已经退出。系统的中断处理程序将负责清理当前进程的资源并选择新的进程运行。

函数功能：结束当前进程的执行，并通知系统当前进程已经退出。

## 4. 实验结果

---



```
QEMU
Father Process: Ping 1, 7;
Child Process: Pong 2, 7;
Father Process: Ping 1, 6;
Child Process: Pong 2, 6;
Father Process: Ping 1, 5;
Child Process: Pong 2, 5;
Father Process: Ping 1, 4;
Child Process: Pong 2, 4;
Father Process: Ping 1, 3;
Child Process: Pong 2, 3;
Father Process: Ping 1, 2;
Child Process: Pong 2, 2;
Father Process: Ping 1, 1;
Child Process: Pong 2, 1;
Father Process: Ping 1, 0;
Child Process: Pong 2, 0;
```