

# lab3 实验报告

姓名：刘佳璇

学号：231220105

邮箱：[13573529579@163.com](mailto:13573529579@163.com)

## 实验进度

我完成了所有内容，包括选做wait。

## 实验结果

```
QEMU - Press Ctrl-Alt to exit mouse grab
=====
TEST FOR BASIC=====
Parent Process (pid:1): Ping 1, 7;
Child Process (pid:2): Pong 2, 7;
I'm the user idle!
First fork: First entering the idle process
Parent Process (pid:1): Ping 1, 6;
Child Process (pid:2): Pong 2, 6;
Parent Process (pid:1): Ping 1, 5;
Child Process (pid:2): Pong 2, 5;
Parent Process (pid:1): Ping 1, 4;
Child Process (pid:2): Pong 2, 4;
Parent Process (pid:1): Ping 1, 3;
Child Process (pid:2): Pong 2, 3;
Parent Process (pid:1): Ping 1, 2;
Child Process (pid:2): Pong 2, 2;
Parent Process (pid:1): Ping 1, 1;
Child Process (pid:2): Pong 2, 1;
Parent Process (pid:1): Ping 1, 0;
Child Process (pid:2): Pong 2, 0;
=====
TEST FOR BASIC OVER=====
```

```
QEMU - Press Ctrl-Alt to exit mouse grab
=====
TEST 1 FOR WAIT=====
Child Process (pid:2, ppid:1): Pong 2, 3;
Child Process (pid:2, ppid:1): Pong 2, 2;
Child Process (pid:2, ppid:1): Pong 2, 1;
Child Process (pid:2, ppid:1): Pong 2, 0;
first wait() returns: 1
second wait() returns: 0
Parent Process (pid:1, ppid:0): Ping 1, 3;
Parent Process (pid:1, ppid:0): Ping 1, 2;
Parent Process (pid:1, ppid:0): Ping 1, 1;
Parent Process (pid:1, ppid:0): Ping 1, 0;
=====
TEST 1 FOR WAIT OVER=====

=====
TEST 2 FOR WAIT=====
Parent Process (pid:1, ppid:0): Ping 1, 3;
Parent Process (pid:1, ppid:0): Ping 1, 2;
Parent Process (pid:1, ppid:0): Ping 1, 1;
Parent Process (pid:1, ppid:0): Ping 1, 0;
Child Process (pid:2, ppid:0): Pong 2, 3;
Child Process (pid:2, ppid:0): Pong 2, 2;
Child Process (pid:2, ppid:0): Pong 2, 1;
Child Process (pid:2, ppid:0): Pong 2, 0;
=====
TEST 2 FOR WAIT OVER=====
```

## 实验修改的代码位置

1. lib/lib.h: 添加定义

- #define SYS\_GETPPID 6
- #define SYS\_WAIT 7
- pid\_t getppid();
- int wait();

2. lib/syscall.c: 添加函数实现

- pid\_t fork()
- int sleep(uint32\_t time)
- int exit()
- pid\_t getpid()
- pid\_t getppid()
- int wait()

3. kernel/kernel/irqHandle.c: 添加函数实现

- void timerHandle(struct StackFrame \*sf);
- void syscallHandle(struct StackFrame \*sf);

- void sysFork(struct StackFrame \*sf);
- void sysExec(struct StackFrame \*sf);
- void sysSleep(struct StackFrame \*sf);
- void sysExit(struct StackFrame \*sf);
- void sysGetPid(struct StackFrame \*sf);
- void sysGetPpid(struct StackFrame \*sf);
- void sysWait(struct StackFrame \*sf);
- uint32\_t schedule();
- void contextSwitch(int nxt); // 上下文切换