

# OS Lab5 Report

name: 刘乐奇

sid: 12011327

Ubuntu用户名: lynchrocket

## 1. 代码中如何区分父子进程？父子进程的执行顺序是否是固定的？

子进程的pid一般是父进程的pid+1；fork()的返回值，在父进程中是子进程的pid，在子进程中是0。父子进程的执行顺序要看CPU调度，不是固定的。

## 2. 请回答第四步僵尸进程中列举的第4种情况的结果会是什么。

父进程不执行wait()，父进程比子进程先结束。在父进程结束时，子进程会过继给继父进程（init进程或者祖父进程），由继父进程来回收未来结束的子进程。

## 3. 请编写一段c语言代码（截图），用于产生僵尸进程，并截图僵尸进程的状态(ps)。

C语言代码如图

```
Welcome C zombie.c X
C zombie.c > main()
1  #include<unistd.h>
2  #include<stdio.h>
3  int main(){
4      if(!fork()){
5          printf("I'm child process.\n");
6      }else{
7          printf("I'm parent process.\n");
8          while(1);
9      }
10     return 0;
11 }
```

僵尸进程的状态如图

```
lynchrocket@lynchrocket-virtual-machine:~/Desktop/OSlab/5$ ps -al
F S  UID      PID     PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S   1000      1109      1105  0  80   0  - 119427 ep_pol  tty2        00:00:20 Xorg
0 S   1000      1322      1105  0  80   0  - 47831  do_sys  tty2        00:00:00 gnome-sess
0 R   1000      6003      5834 96  80   0  -   624  -      pts/1        00:00:25 zombie
1 Z   1000      6004      6003  0  80   0  -    0  -      pts/1        00:00:00
0 R   1000      6007      5657  0  80   0  -   3622 -      pts/0        00:00:00 ps
lynchrocket@lynchrocket-virtual-machine:~/Desktop/OSlab/5$
```

## 4. lab5的ucore代码具体通过哪条指令以什么形式跳转至init\_main()

在 /kern/process/proc.c 的 proc\_init() 函数中调用

了 `int pid = kernel_thread(init_main, "Hello world!!", 0)`。在这个函数里将 `init_main()` 函数的地址复制到中断帧的 `s0` 寄存器中，并在最后调用了 `do_fork()` 函数创建新进程。

之后在 /kern/init/init.c 中调用了 `cpu_idle()` 函数，其中调用了 `schedule()` 函数，在这个函数中切换进程。

由于在初始化的时候对中断帧做了一点手脚，`epc` 寄存器指向的是

`kernel_thread_entry`。在 `kernel_thread_entry` 中跳转到 `s0` 寄存器保存的地址，也即 `init_main()` 函数的地址。

## 5. lab5的ucore代码中是如何调用forkret和forkrets的，forkrets的具体功能是什么？

在 /kern/init/init.c 中调用了 proc\_init() 函数。在这个函数里通过 int pid = kernel\_thread() 调用了 kernel\_thread() 函数。在这个函数里设置了中断帧的一些参数，并在最后调用了 do\_fork() 函数创建新进程。

之后在 /kern/init/init.c 中调用了 cpu\_idle() 函数，其中调用了 schedule() 函数，在这个函数中调用了 proc\_run() 函数，其中调用 switch\_to() 函数来进行上下文切换。switch\_to() 函数中保存了寄存器中的值，然后跳转到ra寄存器保存的地址。由于之前ra寄存器中保存了 forkret() 函数的地址，所以此处会调用 forkret() 函数。forkret() 函数中调用了 forkrets() 函数。

forkrets() 函数把传进来的参数，也就是进程的中断帧放在了sp寄存器中，这样在 \_\_trapret 中就可以直接从中断帧里面恢复所有的寄存器。