Linux进程创建与销毁可视化

陆宇霄 同组：张延辞 张童

进程信息可视化

利用pyQt进行可视化

在有进程创建的时候，通过父子进程的关系形成树状结构。并且显示父子进程之间进程控制块等数据的关联与不同

在有进程销毁的时候，破坏这个树形结构，并且重构

sys_fork:

call _find_empty_process

.....

.....

.....

call _copy_process

添加日志，在调用此函数的时候输出信息提示此函数已经被调用

添加日志，在调用此函数的时候输出信息提示此函数已经被调用

```
int  find_empty_process (void)
{
  int i;

repeat:
  if ((++last_pid) < 0)
    last_pid = 1;
  for (i = 0; i < NR_TASKS; i++)
    if (task[i] && task[i]->pid == last_pid)
      goto repeat;
  for (i = 1; i < NR_TASKS; i++) // 任务0 排除在外。
    if (!task[i])
      return i;
  return -EAGAIN;
}
```


输出这个i可以得到开机以来累计的进程数，这个i就是新创建的进程的任务号

Copy_process 比较长，总共六个关键帧：

- 为子进程创建task_struct，将父进程的task_struct复制给子进程。
- 为子进程的task_struct,tss做个性化设置。
- 为子进程创建第一个页表，也将父进程的页表内容赋给这个页表。
- 子进程共享父进程的文件。
- 设置子进程的GDT项。
- 最后将子进程设置为就绪状态，使其可以参与进程间的轮转。

输出这两步，task_struct中改变了的变量值，进行前后对比

p->state = TASK_RUNNING; 输出前后状态

进程销毁主要有四个关键帧:

- 调用free_page_tables去释放代码段和数据段，get_base通过ldt[1]和ldt[2]这两个描述符来找到代码段和数据段的起始位

输出ldt[1] ldt[2]来确定销毁的进程的代码段和数据段在哪

- do_exit()之后释放了绝大多数占用的资源，但是进程描述符表和内核堆栈共用的那块地址空间没有释放

- sys_waitpid()当pid>0表示等待回收该pid的子进程；pid=0时，回收进程组号等于当前进程号的子进程；pid<-1时回收进程号为-pid的子进程；pid=-1，回收任何子进程。

输出pid

- option可以使该系统调用直接返回或者陷入阻塞等待；stat_addr用来保存状态信息。

输出stat_addr