**操作系统 hw3**

**陈彦帆 2018K8009918002**

一.

1. 用结构体表示，存在proc.h文件中：

struct proc {

  uint sz;                     // Size of process memory (bytes)

  pde\_t\* pgdir;                // Page table

  char \*kstack;                // Bottom of kernel stack for this process

  enum procstate state;        // Process state

  int pid;                     // Process ID

  struct proc \*parent;         // Parent process

  struct trapframe \*tf;        // Trap frame for current syscall

  struct context \*context;     // swtch() here to run process

  void \*chan;                  // If non-zero, sleeping on chan

  int killed;                  // If non-zero, have been killed

  struct file \*ofile[NOFILE];  // Open files

  struct inode \*cwd;           // Current directory

  char name[16];               // Process name (debugging)

};

2.

阅读源码proc.c和exec.c：

fork(): 在进程表里寻找一个未使用的进程，若找到，创建一个新进程，将父进程的信息拷贝给子进程：包括proc结构体中除了chan, killed, context的其它所有信息。

wait(): 在进程表里顺序查找是否有当前进程的子进程。若未找到，返回-1。若找到，等待子进程进入ZOMBIE状态，然后修改子进程的kstack, pgdir, pid, parent, name[0], killed均为0, state=UNUSED，然后返回子进程原来的pid。

exec(): 加载对应的elf文件，创建用户栈，压入0xffffffff作为伪返回PC，压入参数，之后更新当前进程的以下信息：

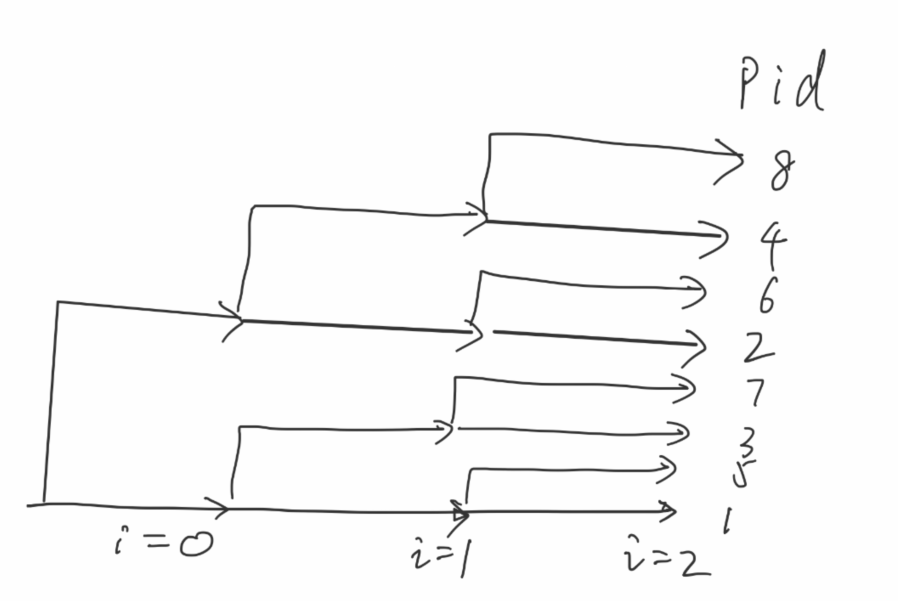
  curproc->pgdir = pgdir;

  curproc->sz = sz;

  curproc->tf->eip = elf.entry;  // main

  curproc->tf->esp = sp;

之后跳转到程序入口。

二. 共8个进程。生成关系图如下（pid不唯一）

hw4

源代码如下：

#include <pthread.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#define M 10000000

static inline unsigned long rdtsc(void)

{

    unsigned long tickl,tickh;

    asm volatile   (

        "rdtsc\n\t"

        :"=a"(tickl),"=d"(tickh));

    return ((unsigned long )tickh<<32)|tickl;

}

int a2 = 0;

void \*thread\_add(void \*arg)

{

    int tmp = 0;

    for (int i = 1; i < M; i+=2)

    {

        tmp += ((int \*)arg)[i];

    }

    a2 = tmp;

    return NULL;

}

int main(void)

{

    unsigned long tick1,tick2;

    int a1 = 0;

    int \* sum = (int \*)malloc(M\*sizeof(int));

    memset(sum,-1,M\*sizeof(int));

    pthread\_t mythread;

    tick1 = rdtsc();

    if (pthread\_create(&mythread, NULL, thread\_add, sum))

    {

        printf("error creating thread.");

        abort();

    }

    for (int i = 0; i < M; i+=2)

    {

        a1 += sum[i];

    }

    if (pthread\_join(mythread, NULL))

    {

        printf("error joining thread.");

        abort();

    }

    int result = a1 + a2;

    tick2 = rdtsc();

    printf("Result is: %d\nTime is %lu clocks\n", result,tick2-tick1);

    return 0;

}

编译指令：

gcc -o t1 t1.c -lpthread

运行结果：

双核虚拟机（linux 64）：

Result is: -10000000

Time is 47094555 clocks

单核虚拟机：

Result is: -10000000

Time is 375961555 clocks

加速比：7.98